VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Off-line lokalizace pomocí rádiových sítí na platformě Android

Android-based Off-line Radio Network-based Localization

2016

Peter Gorčák

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání diplomové práce

Student: **Bc. Peter Gorčák**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T059 Mobilní technologie

Téma: Off-line lokalizace pomocí rádiových sítí na platformě Android
Android-based Off-line Radio Network-based Localization

Jazyk vypracování: čeština

Zásady pro vypracování:

Vytvořte aplikaci, která bude využívat off-line databázi základnových stanic a WiFi AP k hrubé lokalizaci polohy (včetně zpřístupnění polohy aplikacím pomocí vlastní implementace LocationProvider). Pro lokalizaci využijte několik známých algoritmů (v závislosti na tom, jaké informace jsou na konkrétním zařízení k dispozici) a připravte API, umožňující snadnou integraci případných dalších metod.

Aplikace by měla rovněž umožňovat sběr dat v případě, kdy budou k dispozici data o přesné poloze z GNSS (typicky GPS) a poskytovat možnost tyto informace sdílet s dalšími uživateli prostřednictvím serveru.

1. Zjistěte, jaké možnosti lokalizace poskytuje platforma Android, jaké informace o BTS a pokrytí jejich signálem jsou k dispozici z veřejných zdrojů (a v jaké podobě) a zda existují aplikace s podobným či stejným zaměřením.
2. Určete, jakým způsobem budou informace uloženy, sdíleny či získávány klienty (HTTP, cloudové řešení, webové služby, apod.) a jak bude zajištěna jejich dostupnost v off-line režimu. Zdokumentujte formát, v němž budou informace předávány. Popište případné nestandardní knihovny a nástroje, které budete využívat.
3. Prozkoumejte, jaké pokročilejší metody lokalizace na základě informací o buňkách sítě a přístupových bodech existují a navrhněte jejich využití v aplikaci.
4. Analyzujte a navrhněte API pro práci s lokalizačními informacemi, navrhněte a implementujte rozšiřitelnou knihovnu, realizující toto API a ukázkovou mobilní aplikaci, která je bude využívat (a případnou serverovou část).
5. Výsledné řešení otestujte alespoň na dvou různých mobilních zařízeních a zhodnoťte dosažené výsledky. Srovnejte funkčnost Vaší aplikace s aplikacemi z bodu 1.

Seznam doporučené odborné literatury:

[1] James Steele, Nelson To, The Android Developer's Cookbook: Building Applications with the Android SDK, Addison-Wesley Professional, 2010, ISBN-13: 978-0321741233
[2] Reto Meier, Professional Android 4 Application Development, Wrox, 2012, ISBN-13: 978-1118102275
[3] Android Developers [online]. [cit. 2013-09-10]. Dostupné z: http://developer.android.com/
[4] Figueiras, João, and Simone Frattasi. Mobile positioning and tracking: From conventional to cooperative techniques. John Wiley & Sons, 2011. ISBN: 978-1-119-95756-0.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Pavel Moravec, Ph.D.**

Datum zadání: 01.09.2014

Datum odevzdání: 29.04.2016

doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

V Ostrave 27. apríla 2016 .....

Súhlasím so zverejnením tejto diplomovej práce podľa požiadavky čl. 26, odst. 9 Štúdijného a skušobného rádu pre štúdium v magisterských programoch VŠB-TU Ostrava.

V Ostrave 27. apríla 2016 .....

Na tomto mieste by som rád poďakoval Ing. Pavelovi Moravcovi, Ph.D. za odbornú pomoc a cenné rady pri tvorbe tejto diplomovej práce.

Abstrakt

Táto diplomová práca sa zaoberá implementáciou rozšíriteľnej knižnice pre offline lokalizáciu mobilného zriadenia na platforme Android. Polohu určuje pomocou mobilnej rádiovej siete alebo WIFI siete. Knižnica podporuje verziu systému Android od 4.0 Ice Cream Sandwitch a vyššie. Táto práca popisuje známe metódy určovania polohy mobilného zariadenia pomocou rádiových sietí, možnosti určovania polohy pomocou Android frameworku, implementáciu samotnej knižnice spolu so spôsobom spracovania dát. Ďalšia časť práce popisuje tvorbu jednoduchej ukážkovej aplikácie, ktorá využíva implementovanú knižnicu a v závere sa nachádzajú výsledky určovania polohy pri rôznych trásách v porovnaní s GPS polohou. Knižnica a aplikácia boli vytvorené pomocou prostredia Android Studio.

Kľúčove slová: API, BTS, cache, lokalizácia, mapa, poloha, signál, rádio, rozhranie, trieda, WIFI

Abstract

This thesis deals with implementation of an expandable library for offline mobile positioning on the Android platform. The position is determined by the cellular or a WIFI network. The library supports Android 4.0 Ice Cream Sandwitch and higher. This thesis describes the known methods for determining the location of device using radio networks, positioning capabilities using the Android framework, the implementation of the library itself, along with the method of data processing. Further, the creation of a simple application that uses created library is described and at the end are positioning results in a variety of routes, compared with the GPS location. Library and applications have been created using Android Studio environment.

Key Words: API, BTS, cache, localization, map, location, signal, interface, radio, class, WIFI

Obsah

Zoznam použitých skratiek a symbolov

API	–	Application Programming Interface, aplikačné rozhranie
AP	–	Access Point, prístupový bod
BTS	–	Base transmitter staion, rádiová základňová stanica
HTTP	–	Hyper Text Transfer Protocol
UI	–	User Interface, užívateľské rozhranie
WIFI	–	Wireless Fidelity, bezdrôtová sieť

Zoznam obrázkov

Zoznam výpisov zdrojového kódu

1 Úvod

Možnosť lokalizovať mobilné zariadenie sa v dnešnej dobe považuje už sa samozrejmosť, no stále to nemusí byť jednoduchá úloha. Hlavne v oblastiach s obmedzenou konektivitou, prípadne žiadnou, môže byť lokalizovanie zariadenie značne obtiažné a nepresné, pričom lokalizačné služby sú už veľmi rozšírené a ich využívanie stále rastie. Spolu s rozmachom mobilných zariadení s rôznými operačnými systémami, rastie aj používanie týchto služieb a stávajú sa súčasťou nielen záchranných a bezpečnostných systémov, no aj bežne používaných aplikácii ako napríklad rôzne navigácie alebo sociálne siete.

V tejto diplomovej práci popisujem tvorbu rozšíriteľnej knižnice pre off-line lokalizáciu pomocou rádiových sietí a WIFI sietí na platforme Android. Pri jej tvorbe bol kladený dôraz a možnosť jej najjednoduchšieho použitia. V kapitole 2 sú popísané spôsoby delenia lokalizačnch metód, ich princípy fungovania a použitie. Kapitola 3 sa venuje popisu a porovnaniu možnosti lokalizácie na platforme Android ktorú poskytuje samotný framework. Kapitola 4 rozoberá návrh a implementáciu dátovej vrstvy, akým spôsobom je možné na platforme Android získavať informácie o BTS a WIFI a ako sú tieto informácie spracovávane a reprezentované v implementovanej knižnici, možnosti importu a exportu týchto informácií, čím poskytuje úvod k popisu implementácie lokalizovania vo vytvorenej knižnici. V kapitole 5 je už detailný popis komponent a súčastí, ktoré majú na starosti určenie polohy. Tieto komponenty sú popisované v poradí priebehu lokalizovania. V kapitole 6 je priblížený spôsob tvorby ukážkovej aplikácie ktorá využíva implementovanú knižnicu, jej možnosti a implementácia možností zdieľania monitorovaných dát. V kapitole 7 sú zhrnuté výsledky lokalizovania polohy mobilného zariadenia pomocou tejto knižnice a popis podobne zameraných aplikáci.

2 Lokalizačné metódy

Táto kapitola popisuje rádiové lokalizačné metódy, používajuce signály rádiových systémov pre nájdenie a lokalizovanie polohy mobilného zariadenia. Tieto metódy sú využívané rôznymi núdzovými, monitorovacími alebo informačnými službami. Vzhľadom na to že rádiové systémy pôvodne neboli navrhnuté na určovanie polohy zariadení, implementácia a realizácia rôznych lokačných metód vyžaduje dodatočné vybavenie pre vykonávania rôznych druhov meraní a výpočtov.

2.1 Rozdelenie lokačných metod

Tieto metódy sa dajú rozdeliť na základe úlohy, ktorú mobilné zariadenie a rádiová sieť zohrávajú pri určení polohy, podľa spôsobu merania signálu a získavania dát alebo podľa dostupnosti informácii ďalšieho hardwaru v dosahu.

2.1.1 Rozdelenie podľa funkčnosti mobilného zariadenia a siete

Tieto metódy sa dajú rozdeliť na základe úlohy, ktorú mobilné zariadenie a rádiová sieť zohrávajú pri určení polohy.

Network-based Lokčné metódy vykonávané sieťou. Pri týchto metódach jedna, alebo niekoľko základňových staníc vykonáva potrebné merania a posielajú namerané výsledky do lokačného centra SMLC, ktoré vypočíta polohu zariadenia. Takáto implementácia lokačných metód má výhodu v tom, že nevyžaduje žiadne dodatočné úpravy na mobilných zariadeniach ktoré pomocou týchto techník treba lokalizovať. Úpravu však vyžaduje samotná sieť, pridaním technického vybavenia na určovanie polohy, ako napríklad už spomenutá jednotka SMLC, alebo LMU. Nevýhodou tohto riešenia je zvýšenie záťaže siete. Mobilné zariadenie musí byť v aktívnom móde, aby bolo možné používať tieto metódy [1, 2].

Mobile-based Lokčné metódy vykonávané mobilným zariadením. Meranie a výpočet polohy vykonáva samotné mobilné zariadenie. Tieto metódy dovoľujú zisťovanie polohy aj v IDLE móde pomocou merania kontrolných kanálov ktoré kontinuálne vysielajú. Niektoré dodatočné informácie môže mobilné zariadenie potrebovať zo siete, napríklad poloha základňovej stanice. Výhoda spočíva v tom, že používateľ sám může rozhodnúť kedy svoju polohu poskytnúť [2]. Nevýhoda týchto metód je v tom, že nepodporujú niektoré staršie typy mobilných zariadení [1].

Mobile-assisted Lokačné metódy vykonávané sieťou s asistenciou mobilného zariadenia. Meranie vykonáva mobilné zariadenie, no výsledky odosiela do lokačného centra siete, ktoré sa o výpočet polohy postará. V prípade potreby výsledky odosiela spať mobilnému zariadeniu. Nevýhodou toto riešenia je možné oneskorenie z dôvodu preposielania nameraných a vypočítaných dát medzi mobilným zariadením a lokačným centrom [1, 2].

2.1.2 Rozdelenie podľa zpôsobu merania

Lokačné metódy sa dajú rozdeliť aj na základe spôsobu merania a spracovávania signálu a dát.

Multilateralne Niekoľko základňových staníc zároveň, alebo skoro zároveň vykonáva meranie. Patria medzi lokačné techniky vykonávané sieťou [1].

Unilateralne Mobilná stanica vykonáva meranie signálov od viacerých základňových staníc, preto tieto metódy spadajú do kategórie lokačných techník vykonávaných mobilným zariadením. Tieto techniky sú výhodnejšie vzhľadom na zaťaženie a kapacitu signálu [1].

Bilateralne Pri tejto technike nie je potrebné vykonávať viaceré merania. Mobilné zariadenie rovnako meria signál iba od jednej základňovej stanice, prípadne jedna základňová stanica meria signál od mobilného zariadenia. Tieto techniky sú vhodné pre rozsiahle vidiecké oblasti, kde je pokrytie zabezpečené iba jednou základňovou stanicou [1].

2.1.3 Rozdelenie podľa komunikácie s prvkami v dosahu

Takéto rozdelenie lokačných metód hovorí o ich vzťahu medzi jednotlivými prvkami zúčastnenými pri lokácii, závislosti, prípadne nezávislosti k ich existencii a vzhľadom na ich vzdialenosť medzi sebou.

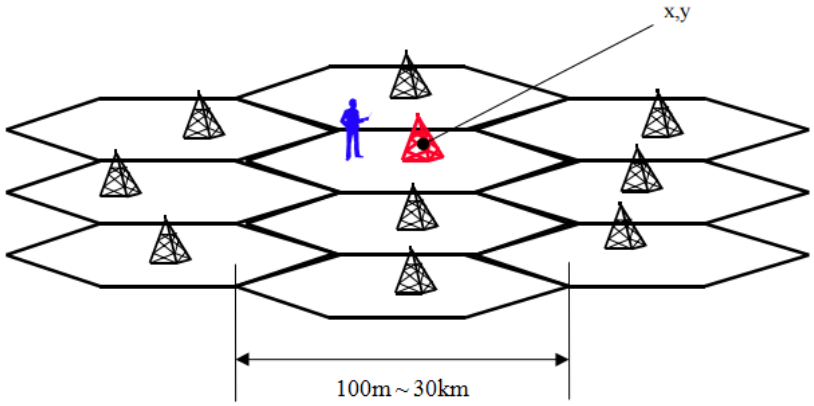
Range-based Tieto metódy sú závislé na rôznych senzorových uzloch, prídavných zariadeniach, ako napríklad časovače, prímače úrovne signálu, smerové alebo sektorové antény, ktoré poskytujú dodatočné informácie a upresňju výsledok určovania polohy. Tieto metódy sa spoliehajú na dostupnosť informácii o vzdialenosti alebo uhle medzi zúčastnenými prvkami, umožňuju pomerne presnú lokalizáci mobilného zariadenia no sú striktne závislé na meraní signálu a časovej synchronizácii [3].

Range-free Metódy spadajúce pod túto kategóriu nie sú závislé na žiadnom dodatočnom hardwari, alebo okolných prvkov v sieti. Plne využívajú vlastnosti siete a jej už existujúceho hardwaru na zistenie polohy pomocou vhodných algoritmov a nie je potreba dodatočných informáciách o vzdialenosti alebo uhlu medzi prvkami v sieti [3].

2.2 Cell ID

Cell ID je najjednoduchšia možná lokačná metoda. Má nízke nároky na implementáciu a nevyžaduje dodatočné úpravy ani na strane mobilného zariadenia ani na strane siete. Patrí medzi bilaterálne spôsoby zisťovania polohy zariadenia a môže byť implementovaná ako network-based alebo aj mobile-based [1]. Je založená na identifikácii základňovej stanice s ktorou je mobilné

zariadenie obsluhované a pomocou ktorej je asociované s určitou bunkou v sieti. Tento identifkátor je prevedený na zemepisnú polohu pomocou informácií o databáze pokrytia od konkrétneho operátora. Prevedenie identifikátora na zemepisnú oblasť je vykoné v jednotke SMLC. Presnosť tejto metódy je striktne závislá od oblasti, ktorú daná bunka pokrýva. Môže sa pohybovať od 300 metrov až po 30 kilometrov. Z toho plynie veľká nevýhoda, ktorou je nepresnosť. V mestských oblastiach, ktoré sú rozdelené na množstvo menších buniek, je tento spôsob ako hrubé zistenie polohy pomerne presný, no vo vidieckých oblastiach, ktoré majú rozlohu niekoľko desiatok kilometrov, je absolútne nespoľahlivá,nakoľko nemusí byť možné určiť smer v akom sa zariadenie od základňovej stanice nachádza. V prípade smerových antén, pokrývajúcich rôzne sektory je táto nevýhoda čiastočne eliminovaná. Výhodou je už spomínaná jednoduchosť a rovnako aj podpora starších zariadení [4]. Na obrázku 1 (prevzaté z [1]) je znázornené rozdelenie siete na bunky obsluhované základňovými stanicami a mobilné zariadenie pridružené konkrétnej stanici. Premenné x a y predstavujú súradnice základňovej stanice.

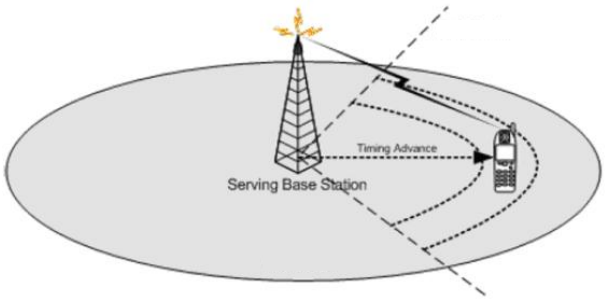


Obr. 1: Určenie polohy pomocou Cell ID

2.3 Timing Advance / Round Trip Time

Tieto metódy umožňujú spresnenie lokalizovania mobilného zariadenia s pouitím metódy Cell ID. Timing Advance sa používa v GSM sietiach. Parameter TA predstavuje približný čas potrebný k tomu, aby signál prešiel od mobilného zariadenia k základňovej stanici. Vďaka tomu určuje približnú vzdialenosť mobilného zariadenia od základňovej stanice a naberá hodnotu od 0 až po 63 s krokom po 550 metrov. V ideálnom prípade využitia tejto metódy je potrebné mať údaje o signále z viacerých rôznych základňových staníc a pomocou jednoduchej triangulácie vypočítať polohu mobilného zariadenia s presnosňou na 550 metrov [4, 2]. Pre LTE siete je presnosť TA približne 78 metrov [5]. Čím je počet staníc v okruhu vyšší, tým je táto metóda presnejšia. Naopak, napríklad pri informáciách o signále od iba jednej základňovej stanice sa pomocou tejto metódy dá určiť len okruh v ktorom sa mobilné zariadenia nachádza. Pri UMTS sietiach sa tejto metóde hovorí Round Trip Time a meria sa ako čas, za ktorý prejde signál od

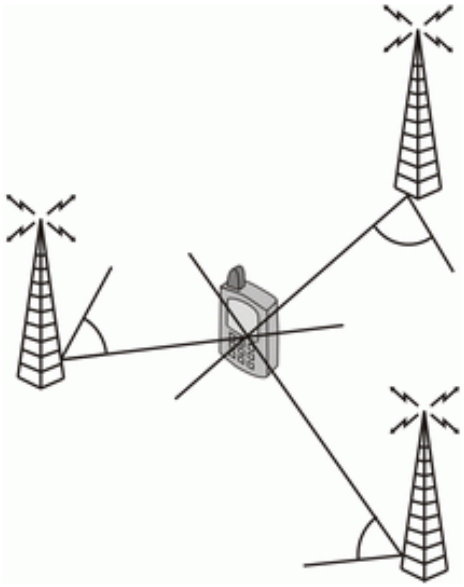
mobilného zariadenia k základňovej stanici a naspäť [4]. Na obrázku 2 je znázornené použitie parametru TA na spresnenie oblasti kde sa mobilné zariadenia nachádza.



Obr. 2: Ukážka parametru TA pri lokalizácii mobilného zariadenia

2.4 Angle of Arrival

Táto metóda používa sústavy smerových antén a znalosti vyžarovacích charakteristík [6]. Poskytuje dostatočnú dvoj-dimenzionálnu geometrickú lokalizáciu mobilného zariadenia na základe uhlu príchodu nameraných dát. Poloha je určená ako priesečník dvoch čiar predstavujúcich smer príchodu signálu smerom od základňových staníc k mobilnému zariadeniu. Čím viac staníc je však v lokalizácii zainteresovaných, tým presnejšie sú výsledky. Na obrázku 3 (dostupné z [6]) ilustračne zobrazuje použitie metódy Angle of Arrival pri lokalizovaní mobilného zariadenia pomocou troch antén.

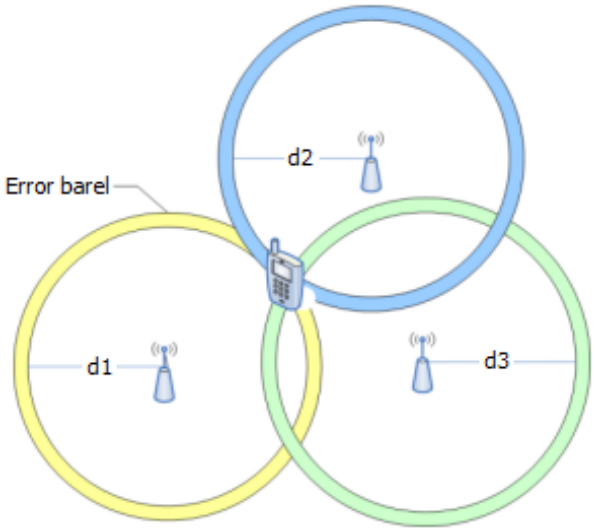


Obr. 3: Metóda Angle of Arrival

Z tohto dôvodu je potrebné, aby bola medzi stanicami priama viditeľnosť, čo nie je jednoduchá požiadavka, hlavne v husto osídlených oblastiach. Vhodnejšie využitie tejto metódy je vo vidieckých oblastiach, vzhľadom na to že k dostačujúcim výsledokm postačia aj namerané dáta z dvoch základňových staníc a interferencia signálu nei je tak vysoká ako v mestských zástavbách [1]. Pri implementácii v 2G sietiach je ďalšou nevýhodou pomerne náročné dodatočné vybavenie, pretože každá základňova stanica potrebuje sústavu smerových antén. Problematické je aj spôsob merania dát, pretože táto technika spadá pod multiraterálne, vyžaduje preto skoro synchronné merania signálov na viacerých základňových staniciach zároveň, čo môže spôsobovať pri veľkom počte mobilných zariadení kapacitné problémy [4, 2].

2.5 Time of Arrival

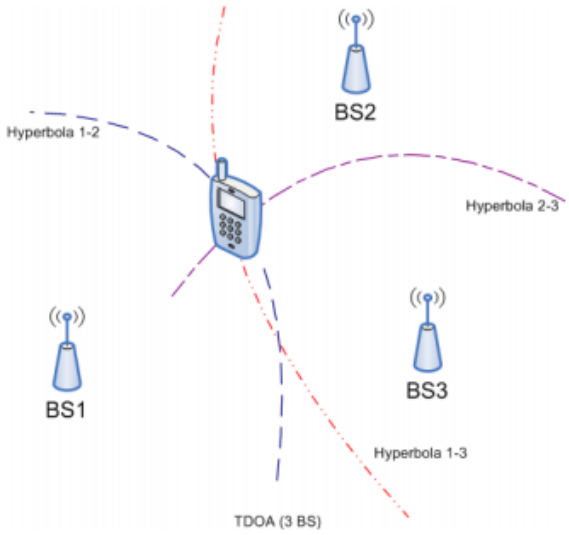
Táto lokalizačná metóda je založená na presnom meraní času príchodu signálu vysielaného z mobilného zariadenia do viac ako jednej základňovej stanice. Vďaka znalosti, že signál cestuje rýchlosťou približne 300 metrov za mikrosekundu, vzdialenosť medzi mobilným zariadením a senzorom môže byť určená na základe uplynutej doby šírenia signálu do cieľa. Tento spôsob vyžaduje aby všetky zúčastnené prvky boli časovo zosynchronizované. Na základe získaných výsledkov je možné určiť rádius okolo každej základňovej stanice, v ktorom sa mobilné zariadenie má nachádzať a pomocou trilaterácie týchto údajov určiť jeho polohu, ako na obrázku 4 (dostupné z [4]). Vzhľadom na rýchlosť šírenia signálu môžu aj malé rozdiely v meraní času, ako napríklad 100 nanosekúnd, spôsobiť nepresnosti okolo 30 metrov [2], čo je v mestských oblastiach s vysokou mierou interferencie a odrazených signálov vysoko pravdepodobné. V porovnaní s metódou Cell ID je presnejšia, no má príliš veľké náklady na úpravu siete [2].



Obr. 4: Princíp metódy Time Of Arrival

2.6 Time Difference Of Arrival

Ide o modifikáciu ToA, no nevyžaduje náročnú časovú synchronizáciu medzi mobilným zariadením a základňovými stanicami účastnenými pri meraní. Časovo synchronizované sú len základňové stanice medzi sebou a každým meraním rozdielu príchodu signálov medzi párom staníc je možné medzi nimi vykresliť hyperbolickú krivku. Meranie na tretej stanici je potrebné kvôli triangulácii a určeniu polohy mobilného zariadenia, ktoré sa nachádza v priesečníku kriviek [4, 2], čo znázorňuje obrázok 5 (dostupné z [4]).



Obr. 5: Time Difference Of Arrival

2.7 Enhanced Observed Time Difference of Arrival

Táto technika bola vyvinutá spoločnosťou Cambridge Position Systems [2] a vyžaduje rozšírenie štruktúry siete o príjmaciu jednotku s názvom Locatiom Measurement Unit, v skratke LMU. Pracuje s meraním času potrebného k prekonaniu vzdialenosti signálom medzi jednotkou LMU a mobilným zariadením, podobne ako pri technike TDoA. Každých 30 sekúnd vysielajú základňpvé stanice asynchronne signál a v prípade, že tento signál je minimálne z troch staníc prijatý mobilným zariadením a jednotkou LMU, je vypočítaný časový rozdiel príchodu týchto signálov na mobilné zariadenie. Vďaka časovým rozdielom je možné vytvoriť pretínajúce sa hyberboly lokalizovať polohu mobilného zariadenia ako ich priesečník. Jedna jednotka LMU dokáže obslúžiť 3 až 5 základňových staníc. Sú to vo svojej podstate upravené mobilné zariadenia, v niektorých prípadoch doplnené aj o GPS modul. Presnosť tejto metódy je závislá na viacerých činiteľov, ako napríklad šum, interferencie, výkonu samotnej LMU jednotky, viac cestnému signálu alebo rozlohe bunky. Presnosť sa pohybuje v rozmedzí od 50 až do 500 metrov s odozvou približne 5 sekúnd [4].

2.8 Enhanced Cell Global Identity

Ide o rozšírenie kombinácie metód Cell ID a Timing Advance o meranie úrovni signálov. Výpočet polohy mobilného zariadenia je rozšírený o model šírenia signálu a na základe nameraných úrovni signálov v oblasti mobilného zariadenia spolu so znalosťami o vysielacích výkonoch základňových staníc sa vypočíta približná poloha mobilného zariadenia. Presnosť metódy je v mestských oblastiach 50 až 550 metrov a pre vidiecké oblasti s menšou hustotou základňových staníc od 250 metrov až 8 kilometrov [6].

2.9 Recieved Signal Strength

Ide o spôsob, ktorý využíva meranie úrovní signálu niekoľkých základňových staníc z kontrolného kanálu na určenie vzdialenosti medzi základňovou stanicou a mobilným zariadením. Za predpokladu použitia všesmerových antén a dodržania podmienok šírenia signálu voľným priestorom je možno okolo každej základňovej stanice vykresliť kružnicu. Pokiaľ sú známe úrovne signálu z aspoň troch základňových staníc, poloha mobilného zariadenia môže byť určená ako priesečník vykreslených kružníc. Podmienky šírenia signálu vo voľným priestorom je však v mestských nemožné dodržať, príjmaná úroveň signálu sa prudko mení, viaccestný signál a jeho tienenie robia veľké problémy, preto by sa mal používať model šírenia signálu v závislosti od prostredia. Okamžitá úroveň signálu sa môže aj na zlomku vlnovej dĺžky rapídne meniť okolo 30 až 40 dB a spôsobovať veľké chyby vo výpočtoch polohy. Táto nevýhoda sa dá čiastočne odstrániť priemerovaním úrovne signálu v čase a pásme. Presnosť techniky veľmi závisí na vhodne zvolenom modeli šírenia signálu v priestore a počte meraní. Podľa spôsobu merania signálu spadá medzi unilaterálne metódy a môžebyť implementovaná ako mobile-based alebo aj ako mobile-assisted. Mobile-based spôsob implementácie vyžaduje, aby základňová stanica vysielala informácie o svojej polohemobilému zariadeniu. Mobilné zariadenie na základe týchto informácie vykonáva merania a posiela ich späť do siete v aktívnom móde. Alternatíva implementácie je taká, že modifikované mobilné zariadenie posiela do siete výpočty aj v prípade že sa nachádza v *idle* móde. Metóda je jednoduchá a rovnako nieje nákladná na určenie polohy zariadenia s použitím iba znalostí o jednej bunke v ktorej sa zariadenie nachádza [1].

2.10 Použitie metód na platforme Android

Na platforme Android je možné určiť hrubú polohu mobilného zariadenia s využitím určitých častí popísaných metód, no s obmedzeniami, pretože framework neposkytuje všetky potrebné informácie a dáta, prípadne operátori niektoré nezverejňujú. Najjednoduchšie je použitie metódy CellID a na základe získaných informácii určiť polohu. Použitie parametru TA je možné iba u siete LTE, pretože pre ďalšie typy sietí framework tento parameter neumožňuje získať. Rovnako nie sú dostupné informácie o AoA alebo ToA, prípadne DTOA alebo EOTDoA, ktoré vyžadujú špecifické upravenie siete. Informácie o RSSI sú dostupné a je teda možnú ich ppužiť pri určovaní polohy. Pri určovaní polohy som sa rozhodol skombinovať informácie o CellID spolu

s informáciami o RSSI. Na základe týchto dát je možné polohu aproximovať napríklad výpočtom ťažiska alebo pomocou metódy váženého priemeru. Pri implementácii lokalizačnej knižnice som sa rozhodol využiť metódu váženého priemeru kvôli univerzálnosti a jej kroky sú detailne popísané v kapitole 5.8.

3 Lokalizácia na platforme Android

V tejo kapitole sú popísané možnosti lokalizácie na platforme Android, ich porvnanie a popis ich dostupných tried, rozhraní, model postupu určovania polohy a informácie, ktoré framework poskytuje o BTS. Operačný systém Android poskytuje pre vývoj aplikácii, ktoré pracujú s polohou zariadenia framework Android location API, ktorý sa nachádza v balíčku `android.location`. Toto API však už Google označil ako nevhodné a doporučuje vývojárom používať *Location Servicies for Android*.

3.1 Android location API

Toto API sa nachádza v balíčku `android.location` a je priamou súčasťou Android frameworku už od verzie Android API 1, teda od samého počiatku. Jeho súčasťou je niekoľko hlavných tried a rozhraní, s ktorými treba pri zisťovaní polohy pracovať. Hlavnou komponentou tohto frameworku je trieda s názvom `LocationManager` [13].

3.1.1 Dostupné triedy

Triedy, ktoré sa v balíčku nachádzajú a poskytujú dôležitú a potrebnú funkcionalitu sú nasledovné:

LocationManager
: Poskytuje prístup k systémovej službe Android location. Táto služba ďalej dáva prístup k různym lokalizačným providerom, k registrácii listenerov pre update lokalizačných dát, proximity upozornení alebo posielanie `Intent` objektov.

LocatonProvider
: Abstraktná trieda, vystupujúca ako nadradená všetkým dostupným typom v API. Periodicky poskytuje informácie o geografickej polohe zariadenia. Každý provider je ďalej definovaný kritériami, podľa ktorých môže byť použitý.

  - Network Poskytuje polohu pomocou rádiovej siete alebo WIFI siete. Je použiteľný aj v uzatvorených priestoroch, napr. budovách.
  - GPS Používa dáta z GPS satelitov na určenie polohy zariadenia. Poskytuje vysokú presnosť, no nie je použiteľný vo vnútri budov.
  - Passive Umožňuje získať informácie o polohe bez spúšťania procesu žiadania lokalizácie. Príjma dáta o polohe z ostatných providerov, ktoré sú žiadané inými aplikáciami. V prípade že nie sú dostupné GPS dáta, může byť veľmi neresný.

Criteria
: Trieda, ktorá upresňuje kritéria aplikácie na výber vhodného providera. Kritéria môžu byť

zoradené podľa presnosti, spotreby batérie, schopnosti poskytovať informácie o altitude, rýchlosti, určovaniu smeru a finančnej náročnosti.

Geocoder
Trieda, ktorá spracúvava geokódovanie a revérze geokódovanie. Geokódovanie vyžaduje backend službu, ktorá poskytuje a spracováva detaily.

Location
Trieda alebo dátova štruktúra reprezentujúca geografickú lokalizáciu. Obsahuje informácie ako sú latitude, longitude, časová známka alebo ďalšie informácie ako napríklad altitude.

Address
Táto trieda reprezentuje adresu v podobe niekoľkých textových reťazcov popisujúcich detaily. Formát je zjednodušená verzia xAL - eXtensible Address Language.

GpsSatellite
Reprezentuje stav aktuálne komunikujúceho GPS satelitu reprezentovaného pomocou triedy `GpsStatus`.

GpsStatus
Popisuje stav konfigurácie GPS satelitu

3.1.2 Dostupné rozhrania

V balíčku[8] sa nachádzajú základné rozhrania slúžiace ako listenery, ktorých implementácia umožňuje zachytávanie callbackov udalostí na ktoré je vzhľadom na potreby aplikácie reagovať:

GpsStatus.Listener
Používa sa na notifikovanie o udalostiach, ktoré popisujú zmenu statusu GPS satelitu. Reagovať možno na štyri typy udalostí:

- `GPS_EVENT_STARTED`
- `GPS_EVENT_STOPPED`
- `GPS_EVENT_FIRST_FIX`
- `GPS_EVENT_SATELLITE_STATUS`

GpsStatus.NmeaListener
Určený pre zachytávanie NMEA dát z GPS. NMEA 0183 je štandard, pôvodne vyvinutý na elektronickú komunikáciu námorníctva a slúži ako bežný formát príjmania dát z GPS satelitov.

LocationListener
Slúži na zaregistrovanie notifikácii o zmene polohy od `LocationManager`.

3.1.3 Základné princípy použitia API

Služby poskytujúce lokalizačné dáta pomocou tohto API je možné využívať pomocou hlavnej komponenty `LocationManager`. Vytvorenie a získanie instancie služby je odporúčané pomocou systémového volania , ktoré vracia handler žiadanej služby na základe parametra, ktorý je v tomto prípade `Context.LOCATION_SERVICE`. Po takomto získaní handlera na instanciu triedy `LocationManager` je možné už pracovať so službami ktoré API poskytuje:

- Získať zoznam dostupných typov `LocationProvider` na získanie a spracovanie poslednej známej polohy zariadenia, ktorá môže byť zároveň aj jeho aktuálna.
- Vytvoriť a zaregistrovať, prípadne odregistrovať periodické aktualizovanie lokalizačných dát pomocou dostupných listenerov a reagovať v aplikácii na zmeni a ich dostupnosť
- Vytvoriť a zaregistrovať, prípadne odregistrovať rôzne typy instancii triedy `Intent`, ktorá slúži ako popis operácii, ktoré je potrebne vykonať alebo ako komunikátor so systémovými službami spustenými na pozadí.

3.2 Location Servicies for Android

Nové API, ktoré sa nachádza v knižnici Google Play Services. Samotný Google preferuje používať toto API oproti predchádzajúcemu API. Poskytuje výkonnejší high-level framework ktorý automaticky spracováva lokalizačné dáta z providerov, používateľov pohyb a presnosť lokalizácie. Poskytuje tiež možnost upravoať notifikovanie lokalizácie, pre potrebu šetrenia batérie zariadenia a veľké množstvo ďalších detailných funkcií týkajúcich sa práce s určovaním polohy zariadenia. Toto API je umiestnené v balíku `com.google.android.gms.location`.

3.2.1 Dostupné rozhrania

V balíku sa nachádza viacero druhov rozhraní ktoré umožňuju pokročilejšie manipulovanie s lokalizáciou mobilného zariadenia alebo slúžia ako komunikačné uzly medzi ďalšími pokročilými súčasťami API:

ActivityRecognitionApi
: Hlavný vstupný bod pre interakciu s aktivitami rozoznávajúcimi aktivity - spôsob pohybu mobilného zariadenia. Slúži ako rozhranie poskytujúce získavať aktuálne informácie o spôsobe pohybu zariadenia.

FusedLocationProviderApi
: Rozhranie poskytuje interakciu s novým typom LocationProvider.

Geofence
: Reprezentuje geografický región a poskytuje možnosť úpravy podľa potreby vyvíjanej aplikácie.

GeofencingApi

Slúži ako vstupný bod komunikácie pre geofencing API.

LocationListener

Používa sa na získavanie notifikácii o zmene polohy zariadenia.

SettingsApi

Umožňuje komunikáciu s API, ktoré spracováva informácie o dostupnosti požadovanej konfigurácie zariadenie aplikáciou.

3.2.2 Dostupné triedy

Triedy obsiahnuté v API poskytujú vačšiu variabilitu možností a funkcionalitu pri určovaní polohy a prácou s ňou:

ActivityRecognition

Prezentuje hlavný bod pri integrácii komponent rozoznávajúcich aktivitu - pohyb zariadenia.

ActivityRecognitionResult

Predstavuje výsledok rozoznávania typu aktivít mobilného zariadenia. Obsahuje zoznam aktivít, ktoré v danom momente používateľ zariadenia pravdepodobne vykonáva.

DetectedActivity

Detekovaná aktivita zariadenia z niekoľkých preddefinovaných, ktoré popisujú predpokladaný spôsob pohybu zariadenia:

- IN_VEHICLE
- ON_BICYCLE
- ON_FOOT
- RUNNING
- STILL
- TILTING
- UNKNOWN
- WALKING

GeofencingEvent

Reprezentuje udalosť vyvolanú v `GeofencingApi`. Môže to byť buď udalosť notifikujúca štart vysielania geofence informácii alebo notifikácia o chybovom stave po zaregistrovaní a ich monitorovaní.

GeofencingRequest

Špecifikuje zoznam geofence informácií, ktoré majú byť monitorované a ako by mali byť notifikované.

LocationAvailability

Predsavuje status dosupnosti a stavu získaných lokalizačných dát, doručených z `LocationCallback`.

LocationCallback

Používa sa na príjimanie notifikácii z `FusedLocationProviderApi` v prípadoch ak sa zaznamená zmena polohy zariadenia. Na získavanie týchto asynchronných udalostí musí byť zaregistrovaný spolu s lokalizačným klientom.

LocationRequest

Dátová štruktúra, popisujúcich kvalitu lokalizačných služieb ožadovaných od `FusedLocationProviderApi`.

LocationResult

Dátová štruktúra, reprezentujúca získané geografické informácie o polohe.

LocationServices

Hlavný komunikačný uzol pre integráciu ďalších lokalizačných služieb.

LocationSettingsRequest

Špecifikuje typ okalizačných služieb, ktoré klient vyžaduje. Tieto požiadavky sú kontrolované pre optimálne fungovanie všetkých vyžadovaných služieb.

LocationSettingsResult

Výsledok kontroly požiadavok na lokalizačné služby, ktorý indikuje či je potrebné zmeniť nastavenia.

LocationSettingsStates

Ukladá stav lokalizačných nastavení.

API obsahuje niekoľko ďalších tried zabezpečujúcich nevyhnutnú funkcionalitu alebo reprezentujúcich detailné informácie: `Geofence.Builder`,`GeofenceStatusCodes`, `GeofencingRequest.Builder`,`LocationSettingsRequest.Builder`, `LocationSettingsStatusCodes` a `LocationStatusCodes`.

3.3 Model lokalizácie

Pri určovaní polohy na platforme Android je dôležité správne určiť priority vzhľadom k požiadavkam a možnostiam.

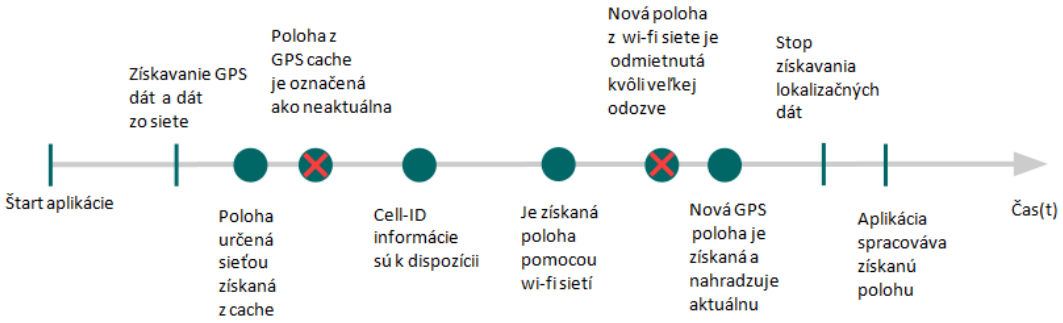
Pri výbere zdroja lokalizačných dát možno použiť dáta z GPS satelitov, ktoré poskytujú veľmi presný výsledok, no majú negatívny dopad na spotrebu batérie zariadenia, nie sú použiteľné vo vnútri budov a samotné získanie polohy môže trvať dlhšie ako je potrebné. Alternatívny zdroj v podobe informácii o signáloch z bunkovej rádiovej sietealebo WIFI siete funguje rovnako vo vnútri budov ako aj v otvorenej krajine, má menší dopad na spotrebu batérie zariadenia, no nemusí poskytovať takú presnosť určenia polohy ako GPS. Zdroje sa môžu používať samostatne alebo kombinovane. Ďalším dôležitým faktorom je pohyb mobilného zariadenia, kvôli čomu je potreba často polohu znova zisťovať. Veľkú úlohu hrá aj konzistentnosť dát. Poloha zariadenia získaná z jedného zdroja, ktorá je považovaná za už neaktuálnu, môže byť stále presnejšia ako poloha získana z iného zdroja a je vyhodnotená ako aktuálna. Kvôli tejto variácii zdrojov, výkonu a presnosti je porebné pri určovaní polohy vytvoriť konzistentný model, ktorý má minimalizovať negatívne vplyvy a poskytovať najlepšie možné výsledky a mal by určovať kedy je potrebné získavať dáta a kedy nie. Pre Android je takýto odporúčaný model vytvorený.

3.3.1 Postup určovania polohy

Typický postup pri získavaní polohy je definovaný nasledovne:

1. Štart aplikácie.
2. Po spustení zaregistrovanie Listener-ov pre update informácií z vyžadovaných LocationProvider-ov
3. Filtrovaním informácií podľa priorít udržiavať najlepší možný odhad polohy zariadenia.
4. Ukončiť a odregistrovať vytvorené Listener-y na získavanie informácií kvôli šetreniu batérie.
5. Pri výpočte polohy použiť informácie o poslednej známej polohe zariadenia.

Na obrázku 6 je na časovej osi vizualizovaný časový úsek, počas ktorého aplikácia získava informácie o polohe zariadenia.



Obr. 6: Časová os, reprezentujúca získavanie informácií o polohe.

3.4 Dostupné informácie o BTS

Android API poskytuje informácie o základňových staniciach rádiových sietí prostredníctvom triedy `TelephonyManager`. Informácie o aktuálne slúžiacej uákladňovej stanici poskytuje metóda `getCellLocation()`. V prípade že ide o GSM mobilné zariadenie, prípadne je pripojené do GSM siete, návratový typ je objekt typu `GsmCellLocation`, ktorý poskytuje informácie ako Cell ID, Location Area Code a Primary Scrambling Code v prípade UMTS siete. Ak mobilné zariadenie podporuje CDMA siete, návratovým typom je objekt `CdmaCellLocation`, ktorý obsahuje viac informácií. K dispozícii je ID základňovej stanice, jej poloha definovaná súradnicami latitude a longtitude, ID siete a ID systému.

`TelephonyManager` poskytuje aj ďalšie detailné informácie, ako napríklad type siete, názov operátora, stav a typ zariadenia, mobile network code, mobile country code a mnoho ďalších.

Informácie o susedných základňových staniciach sú dostupné tiež prostredníctvom triedy `TelephonyManager`, no líšia sa podľa API verzie systému. Pre zariadenia, ktorých API verzia systému je nižšie ako 17, je zoznam susedných základňových staníc získavaný volaním metódy `getNeighboringCellInfo()`, ktorej návratový typ je `List`, obsahujúci objekty typu `NeighboringCellInfo`. Tento spôsob je však problematický a často metóda vráti `null`, vďaka čomu je v mnohých prípadoch nespoľahlivá. Toto je častý problém pre staršie zariadenia a pre vývoj aplikácie je prakticky nevyriešiteľný a od API verzie 23 je táto metóda označená ako *deprecated*. Z tohto dôvodu je v API verzii 17 a vyššie k dispozícii novšia a spoľahlivejšia metóda `getAllCellInfo()`, ktorej návratový typ je tiež `List`, no informácie o jednotlivých základňových staniciach sú už reprezentované inou triedou - `CellInfoItem`. V tomto zozname sa nachádza ako aktuáne slúžiaca stanica, tak aj susedné stanice [7].

3.4.1 NeighboringCellInfo

Reprezentuje informácie o základňovej stanici pre verzie API operačného systému nižšie ako 17. Poskytuje informácie ako Cell ID, LAC, PSC a typ siete. Informácie o sile signálu - RSSI sú dostupné pre GSM ako level *asu* rozsah v hodnote od 0 do 31, kde hodnota 0 znamená -113 dBm alebo menej a hodnota 31 znamená -51 dBm alebo viac, s použitím vzorca $dBm = -113 + 2 * asu$. Pre UMTS táto hodnota predstavuje index levelu CPICH RSCP [11].

3.4.2 CellInfo

Abstraktná trieda slúžiaca ako rodičovská trieda pre ostatné triedy reprezentujúce základňové stanice podľa typu siete - `CellInfoGsm`, `CellInfoCdma`, `CellInfoWcdma` (pribudlo v API verzii 18) a `CellInfoLte`. Každá z tried poskytuje metódu s názvom `getCellIdentity()`, ktorá vracia objekt, ktorého typ reprezentuje identitu základňovej stancie podľa je príslušnosti k sieti [7].

CellIdentityGsm

Poskytuje ID stanice, LAC, MCC, MNC a PSC, ktoré je v tomto prípade nedefinované a vracia hodnotu `Integer.MAX_VALUE`

CellIdentityCdma

Poskytuje ID stanice, ID siete, ID systému, latitude a longitude polohu stanice.

CellIdentityWcdma

Poskytuje ID stanice, LAC, MCC, MNC a 9-bitový UMTS PSC.

CellIdentityLte

Poskytuje ID stanice, MCC, MNC, TAC a PCI (ID fyzickej vrstvy stanice). PCI nadobúda hodnotu od 0 po 503 a slúži na zakódovanie dát za účelom ich odlíšenia sa od dát z ostaných staníc.

Informácie o pokrytí signálom základňovej stanice sú k dispozícii prostredníctvom metódy `getCellSignalStrength()`, ktorej návratový typ je tiež spätý s príslušnosťou bunky k typu siete.

CellSignalStrengthGsm

Poskytuje asu level signálu v rozsahu 0 až 31 a silu signálu v jednotkách dBm.

CellSignalStrengthCdma

Poskytuje asu level signálu v rozsahu 0 až 97, RSSI hodnotu v dBm, Ec/Io pomer v jednotkách dB*10, silu signálu v dBm, EVDO RSSI hodnotu v dBm, EVDO Ec/Io hodnotu v dB*10 a pomer signálu nosnej k rušeniu.

CellSignalStrengthWcdma

Poskytuje asu level signálu v rozsahu 0 až 31 a silu signálu v jednotkách dBm.

CellSignalStrengthLte

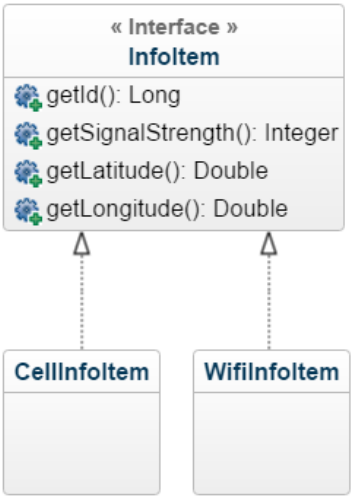
Poskytuje asu level LTE signálu v rozsahu 0 až 97, silu signálu v jednotkách dBm a ako jediná, poskytuje informáciu o hodnote TA

4 Návrh a implementácia dátovej vrstvy

Táto kapitola popisuje návrh a implementáciu dátovej vrstvy implementovanej knižnice. Ďalej popisuje spôsob, akým sú spracovávane namerané dáta o základňových staniciach a prístupových bodoch WIFI v implementovanej knižnici spolu problémami spôsobenými podporou aj staršej verzie systému Android a akým modelom sú tieto informácie spracovávane. Neskôr v ďalších častiach sa kapitola venuje popisu návrhu a spracovávania zápisu dát do logu pre ďalšie použitie.

4.1 Rozhranie InfoItem

Vzhľadom na rozdielnosť informácii o BTS medzi verziami API 16 a menej a verziami s API 17 a vyššie je v knižnici vytvorený spoločný jednotný typ pre jednoduchšie spracovávanie týchto informácii. Rovnako aj pre informácie o WIFI je vytvorený vlastný typ pre ich ďalšie spracovávanie. Vďaka tomu sú ukladané len informácie potrebné na výpočet polohy zariadenia a aj samotný proces ukladania a získavania dát je jednoduchší. Oba typy informácii sú teda reprezentované odlišnými triedami, pre ktoré je základ rozhranie `InfoItem`, ktoré musí každá trieda v knižnici implementovať, čím je zaručená určitá základná jednotnosť pri práci s nimi, kvôli nutnosti implementácie základných metód. Toto rozhranie je implementované triedami `CellInfoItem` a `WifiInfoItem`, čo je znázornené na diagrame tried na obrázku 7.



Obr. 7: Diagram tried použitia rozhrania InfoItem.

Z diagramu je jasné, že triedy musia implementovať štvoricu základných metód rozhrania. Obsah tried `CellInfoItem` a `WifiInfoItem` v diagrame nie je znázornený kvôli čitateľnosti, pretožetriedy obsahujú množstvo špecifických atribútov a metód.

4.1.1 Trieda CellInfoItem

Touto triedou sú reprezentované informácie o BTS pre všetky verzie API. Táto trieda obsahuje nasledovné atribúty:

- `int type` - inforuje o type stanice: GSM, WCDMA, CDMA, LTE
- `int id` - id stanice
- `double lat` - latitude, súradnica zemepisnej šírky polohy stanice
- `double lon` - longitude, súradnica zemepisnej dĺžky polohy stanice
- `int lac` - Location Area Code
- `int mnc` - Mobile Network Code
- `int mcc` - Mobile Country Code
- `int psc` - Primary Scrambling Code
- `int system id` - ID systému, pre CDMA siete
- `int network id` - ID siete, pre CDMA siete
- `int signalStrength_dBm` - sila signálu v jednotkách dBm, použitá pri výpočte polohy
- `int located` - informuje o tom či daná stanica je už lokalizovaná alebo nie

4.1.2 Trieda WifiInfoItem

Touto triedou sú reprezentované informácie o WIFI. Táto trieda obsahuje nasledovné atribúty:

- `long bssidLong` - SSID konvertované na číslo long slúžiace ako ID pre ukladanie v SQLite databáze
- `String bssid` - BSSID prístupového bodu
- `String ssid` - SSID prístupového bodu
- `double lat` - latitude, súradnica zemepisnej šírky polohy prístupového bodu
- `double lon` - longitude, súradnica zemepisnej dĺžky polohy prístupového bodu
- `String macAdress` - MAC adresa prístpového bodu
- `int located` - informuje o tom či daná stanica je už lokalizovaná alebo nie
- `int rssi` - sila prijatého signálu v jendotkách dBm

4.2 Získavanie informácii o BTS

Dáta o BTS sú získavané pomocou triedy `TelephonyManager`. Táto trieda bola bližšie spolu s problémom rôzných verzií API popísaná v kapitole 3.3.1 Vzhľadom na rozdielnosť vo verziách API, je nutné skontrolovať verziu zariadenia a následne získať informácie o BTS. V ďalšom kroku sú tieto dáta pomocou triedy `LocationItemCreator` a jej statických metód použité na vytvorenie zoznamu `List`, ktorý obsahuje instancie nezávislej triedy `CellInfoItem`, s ktorými sa už ďalej v knižnici pracuje jednotne, bez ohľadu na verziu API zariadenia.

V prípade najhoršej možnej varianty, kedy je použité zariadenie s API verziou systému menej ako 17 a informácie o susedných základňových staniciach nie je možné získať je vo výpočte a určení polohy použitá iba stanica s ktorou mobilné zariadenie komunikuje. V takomto prípade je presnosť určenej polohy určená iba polohou samotnej stanice, čo je veľmi nepresný údaj, pretože kvôli obmedzeniam API systému Android nie je možné zistiť ani vyžarovací výkon základňovej stanice nazáklade torého by bolo možné čiastočne určiť priemer oblasti v ktorej sa mobilné zariadenie nachádza. Časť kódu, ktorá tieto informácie získava a spracováva ich, sa nachádza vo výpise kódu 1:

```
List <CellInfoItem> cellsInRange;
if(Build.VERSION.SDK_INT >= 17){
    List<CellInfo> cellInfoList = telephonyManager.getAllCellInfo();
    cellsInRange = LocationItemCreator.createCellItems(cellInfoList) ;
}
else{
    List<NeighboringCellInfo> neighborsInfo = telephonyManager.
       getNeighboringCellInfo();
    cellsInRange = LocationItemCreator.createCellItemsOldAPI(neighborsInfo) ;
}
```

Výpis 1: Získavanie informácii o susedných BTS podľa verzie API

4.3 Získavanie informácii o WIFI

Pri získavaní informácii o dostupných WIFI prístupových bodoch je výhoda v jednotnosti API a nie je preto potrebné kontrolovať verziu zariadenia a na jej základe volať rozdielne metódy. Tieto informácie sú získavané prostredníctvom triedy `WifiManager` a následne pomocou triedy `LocationItemCreator` je vytvorený zoznam objektov typu `WifiInfoItem`. Tento spôsob je znázornený na výpise programu 2.

```
List<ScanResult> results = wifiManager.getScanResults();
List<WifiInfoItem> wifiItems =LocationItemCreator.createWifiItems(results);
```

Výpis 2: Získavanie informácií o prístupových bodoch WIFI v okolí

4.4 Logvanie dát

Logovanie dát má na starosti trieda `LocationDataIO` a rozhranie `Loggable`. Toto rozhranie v knižnici slúži pri zápise dát do súboru a tvorbu logovacích dát. Triedy, ktoré ho implementujú, musia implementovať jedinú metódu:

```
String buildLogString(String description, String label)
```

Implementovaním tejto metódy konkrétnymi triedami umožňuje vytvoriť pre ne špecifický reťazec ktorý je následne použitý pri zápise dát. Parametre `description` a `label` slúžia na možné pridanie dodatočných informácií pri vytváraní reťazca. Uplatnenie je demonštrované v prípade exportu informácii o BTS do formátu CLF, kedy je zapisovaný reťazec odlišný ako reťazec nesúci informácie o prístupovom bode WIFI, ktorý je exportovaný do súboru vo formáte CSV. Zápis a čítanie majú už na starosti metódy nachádzajúce sa v triede `LocationDataIO`. Jednotlivé riedy sa od seba líšia vzhľadom na parametre a spôsob ako pripravia dáta, no zápis vykonáva jednotne statická metóda `void writeLogData(...)`. Metóda má osem vstupných parametrov ktoré z dôvodu lepšej čitateľnosti nie sú obsiahnuté v jej deklarácii:

- `Context context` - umožňuje vytvorenei a zobrazenie Toast správy, ktorá na záver informuje o úspechu, prípadne neúspechu zápisu dát.
- `List<Loggable> loggableList` - zoznam objektov, ktorých informácie budú zapísané. Môže ísť napríklad o objekty typu `CellInfoItem` alebo `WifiInfoItem`, pretože obe tieto triedy implementujú rozhranie `Loggable`.
- `String dstFolderName` - definuje názov adresára v ktorom bude výsledný súbor vytvorený. V prípade že je parameter nastavený na hodnotu `null`, je požitý defaultný názov - `OfflineLogData`
- `String fileName` - definuje názov výsledného súboru.
- `String extension` - umožňuje definovať príponu výsledného súboru.
- `String logItemDesc` - ide o parameter umožňujúci pridávať do záznamu dodatočné informácie.
- `String logItemLabel` - rovnaká úloha ako parameter `logItemDesc`.
- `String finishMessage` - správa, ktorá sa zobrazí po dokončení zápisu.
- `boolean append` - prepíše existujúci súbor, prípadne iba pridá dáta

Na zápis do súboru sú použité triedy `FileWriter`, `BufferedWriter` a dáta sú v cykle zapisované. Každý jeden zapisovaný záznam je vytvorený prostredníctvom implementovanej metódy `String buildLogString(String description, String label)`, ktorá je volaná nad každým

objektom v zozname `loggableList`. Samotný zápis prebieha v cykle, ktorý prechádza zoznam objektov na zápis. Parametrom `append` sa pri vytváraní instancie triedy `FileWriter` umožňuje nastaviť spôsob zápisu. Ak je tento parameter nastavený na hodnotu `true`, dáta budú v prípade, ak daný súbor už existuje pridané k jeho obsahu. V prípade že je parameter nastavený na hodnotu `false`, dáta budú zapísané do nového súboru a v prípade, že súbor s požadovaným menom už existuje, bude jeho obsah prepísaný novými dátami. Táto možnosť je využitá v ukážkovej aplikácii, ktorej úlohou je aj ukládať získané dáta, ktoré sú priebežne zapisované po nahromadení minimálneho množstva, pretože nie je potrebné čakať na ukončenie aplikácie. Takto zvolený spôsob rovnako zabraňuje nárazovému zápisu priveľkého množstva získaných dát. Časť kódu starajúca sa o zápis v tejto metóde je zobrazená na výpise 3

```
try {
    BufferedWriter out;
    FileWriter fileWriter= new FileWriter(file);
    out = new BufferedWriter(fileWriter);
    for(Loggable logItem : loggableList){
        String item = logItem.buildLogString(logItemDesc,logItemLabel);
        out.write(item);
        out.newLine();
        out.flush();
    }
    out.close();
    fileWriter.close();
    if(finishMessage!=null)
        Toast.makeText(context,finishMessage+" "+file.getAbsolutePath(),Toast.
            LENGTH_LONG).show();
} catch (IOException e) {
        e.printStackTrace();
}
```

Výpis 3: Zápis log dát

4.5 Formát pre import a export

Knižnica umožňuje import a export informácii o BTS a WIFI za účelom zdieľania týchto informácii. Informácie o BTS je možné importovať a exportovať vo formáte CLF verzia 3.3.0. Informácie o prístupových bodoch WIFI je možné importovať a exportovať vo formáte CSV.

4.5.1 CLF

CLF je formát, ktorý umožňuje štruktúrované ukladanie dát o BTS. Pôvodne sa používal na import týchto údajov v aplikáciach Signal Monitoring a Field Test pre zariadenia Nokia [12]. Formát ukladania dát vyzerá nasledovne:

- MCC + MNC - spolu, bez medzier
- CellId - Cell Identity
- LAC - Location Area Code
- RNC - Radio Network Controller
- Latitude - súradnica zemepisnej šírky polohy stanice
- Longitude - longitude, súradnica zemepisnej dĺžky polohy stanice
- POS-RAT - presnosť koordinátov
- DESC - popis stanice, jednoduchý doplnkový text
- SYS - radio systém, 0 - neznámy, 1 - GSM, 2 - CDMA, 3 - UMTS, 4 - LTE
- LABEL - krátky popisok, maximálne 8 znakov
- AZI - azimut stanice v stupňoch
- HEIGHT - výška stanice nad zemou v metroch
- BW - uhol vyžarovania

Pre polia, ktoré su neznáme je použitá hodnota -1. Vďaka použitiu tohto formátu umožňuje knižnica spolupracovať s inými aplikáciami ktoré podporujú import a export v CLF formáte. Jednou z takýchto aplikácii je G-Mon.

4.5.1.1 Export do CLF Pre export infomácii o BTS slúži statická metóda s názvom `void exportCellsToCLF(Context context)`. Metóda je definovaná ako verejná a statická a jej parameter je objekt `Context`, ktorý je potrebný pri prvotnom načítaní dát z internej SQLite databázy. Po načítaní dát z databázy a ich príprave je prevedený ich zápis pomocou spomínanej metódy `void writeLogData(...)`. V tomto prípade sú predávané parametre `String logItemDesc` a `String logItemLabel` použité v CLF zázname v časti DESC a LABEL ktoré sú priblížené v predchádzajúcej kapitole 4.5.1.

4.5.1.2 Import z CLF

Pre import infomácii o BTS slúži statická metóda s názvom `void importCellsFromCLF(Context context, String fileName)`, ktorá je rovnako definovaná ako verejná a statická a jej parametrami sú objekty `Context`, ktorý je potrebný pri otvorení internej SQLite databázy, do ktorej budú načítané údaje uložené a String reťazec, ktorý reprezentuje názov CLF súboru, z ktoréhu sú dáta čítané. Na čítanie z CLF súboru sú použité triedy `BufferedReader` a `FileReader`. V cykle sa prechádza každý jeden záznam v súbore, parsuje a po tomto kroku je vytvorený objekt `CellInfoItem`, ktorý je následne uložený v internej SQLite databáze.

4.5.2 CSV

CSV je jednoduchý formát určený na ukladanie tabuľkových dát, ktoré sú oddelené bodkočiarkou. Tento formát je použitý na import a export informácii o prístupových bodoch WIFI. Dáta sú ukladané v nasledujúcej forme:

- BSSID
- Latitude
- Longitude
- SSID
- MAC address

4.5.2.1 Export do CSV

Pre export infomácii o WIFI slúži statická metóda s názvom `void exportWifiToCSV(Context context, String fileName)`. Metóda je definovaná ako verejná a statická a jej Parameter je objekt `Context`, ktorý je potrebný pri prvotnom načítaní dát z internej SQLite databázy. Rovnako ako v prípade exportu do CLF je po načítaní dát z databázy a ich príprave prevedený ich zápis pomocou spomínanej metódy `void writeLogData(...)`. Parametre `String logItemDesc` a `String logItemLabel` V tomo prípade nie sú využité.

4.5.2.2 Import z CSV

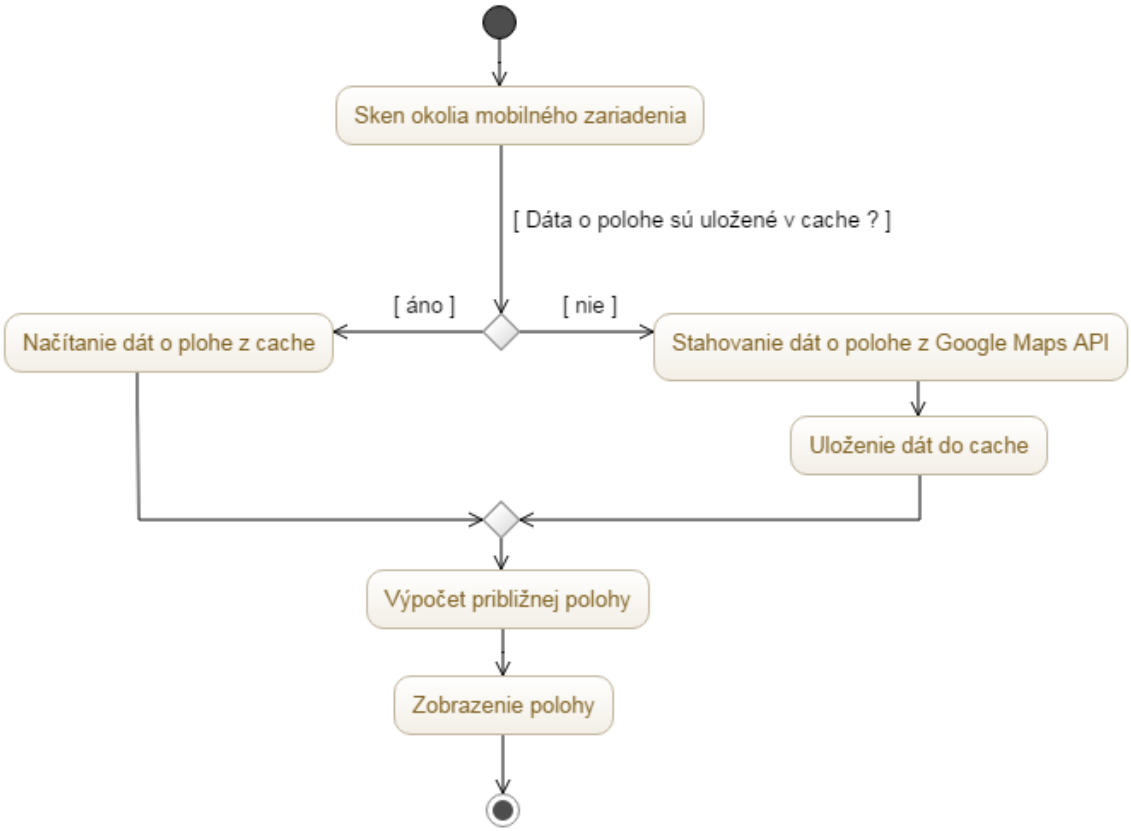
Pre import infomácii o WIFI bodoch slúži statická metóda s názvom `void importWifisFromCSV(Context context, String fileName)`, ktorá je verejná a statická a jej parametrami sú objekty `Context`, ktorý je potrebný pri otvorení internej SQLite databázy, do ktorej budú načítané údaje uložené a String reťazec, ktorý reprezentuje názov CSV súboru, z ktoréhu sú dáta čítané. Na čítanie z CSV súboru sú použité triedy `BufferedReader` a `FileReader`, tak ako pri importe z CLF. V cykle sa prechádza každý jeden záznam v súbore, parsuje a po tomto kroku je vytvorený objekt `WifiInfoItem`, ktorý je následne uložený v internej SQLite databáze.

5 Implementácia lokalizácie

V tejto kapitole je popísaná implementácia knižnice, jej štruktúra, obsah jednotlivých balíčkov, funkcionalita obsiahnutých tried a rozhraní. Pri realizácii knižnice som sa snažil aby bola jednoducho použiteľná a o jej najjednoduchšie riešenie pomocou princípov objektovo orientovaného programovania, aby bolo možné knižnicu v budúcnosti jednoducho rozširvať alebo upravovať. Knižnicu som pomenoval vzhľadom na jej funkciu `OfflineLocation`.

5.1 Priebeh lokalizovania

Pri návrhu priebehu samotného lokalizovania zaradenia bol kladený dôraz na čo najväčšiu jednoduchosť z pohľadu použitia metód knižnice a spracovávania dát. Na obrázku 8 je znázornený diagram aktivít, ktorý reprezentuje priebeh lokaliovania mobilného zariadenia.



Obr. 8: Diagram aktivít zobrazujúc priebeh lokalizovania

V prvom rade je potrebné vykonať sken okolia a zistiť existenciu susedných BTS staníc, prípadne prístupovýchbodoch WIFI a podľa ich dostupnosti rozhodnúť, ktorý typ lokalizácie sa bude ďalej vykonávať. Ďalej je potrebné určiť priebeh určovania lokalizácie. Tento priebeh sa rozdeľuje na dva typy:

- Dáta o susedných BTS staniciach alebo prístupových bodoch WIFI sú už uložené v cache aplikácie. Je teda známa ich poloha a je potrebné iba na základe úrovne signálov z týchto objektov určiť približnú polohu mobilného zariadenia. V tomto prípade knižnica nepotrbuje robiť žiadne HTTP requesty a naozaj si vystačí iba s offline dátami
- Dáta o susedných BTS staniciach alebo prístupových bodoch WIFI sa nenachádzajú v cache aplikácie a pre výpočet polohy je potrebné pomocou HTTP requestov tieto dáta získať a až potom vypočítať polohu mobilného zariadenia. Tieto dáta sú uložené v cache databáze a je možné s nimi už neskôr pracovať v offline režime.

Po získaní dát či už z cache alebo pomocou HTP requestov je potrebné vypočítat na základe úrovni signálov približnú polohu mobilného zariadenia a výsledky poslať na ďalšie spracovanie.

5.2 Štruktúra projektu

Triedy v knižnici sú rozdelené do štruktúry balíčkov podľa ich funkcionality a použiia. Jadro knižnice tvoria 4 hlavné:

- `com.example.gor0020.offlinelocation.location`
- `com.example.gor0020.offlinelocation.log`
- `com.example.gor0020.offlinelocation.persistance`
- `com.example.gor0020.offlinelocation.utils`

Toto rozdelenie je príliš všeobecné a slúži hlavne ako návestie pre rýchlu orientáciu v projekte. Preto každý vymenovaný baíček obsahuje ďalšiu štruktúru, ktorá viac špecifikuje jednotlivé úlohy tried a rozhraní.

5.2.1 com.example.gor0020.offlinelocation.location

Tento balíček obsahuje jadro celej implementovanej knižnice, jej hlavné súčasti, triedy a rozhrania a obsahuje aj najrozsiahlejšiu štruktúru spomedzi ostatných hlavných balíčkov.

5.2.2 com.example.gor0020.offlinelocation.log

Balíček obsahuje triedy a rozhrania používane pri logovaní dát a ich zapisovaní do súborov.

5.2.3 com.example.gor0020.offlinelocation.persistance

V tomto balíčku sa nachádzajú triedy, ktorých úlohou je ukladať informácie o BTS a WIFI, poskytovať k ním prístup, prípadne ich upravovať alebo mazať. Tieto triedy zabezpečujú vytváranie SQLite databázy, jej mazanie alebo upravovanie.

5.2.4 com.example.gor0020.offlinelocation.utils

V tomto balíčku sa nachádzajú triedy, ktorých funkcionalita nie je príliš spätá so žiadnou konkrétnou úlohou a je skôr všeobecná.

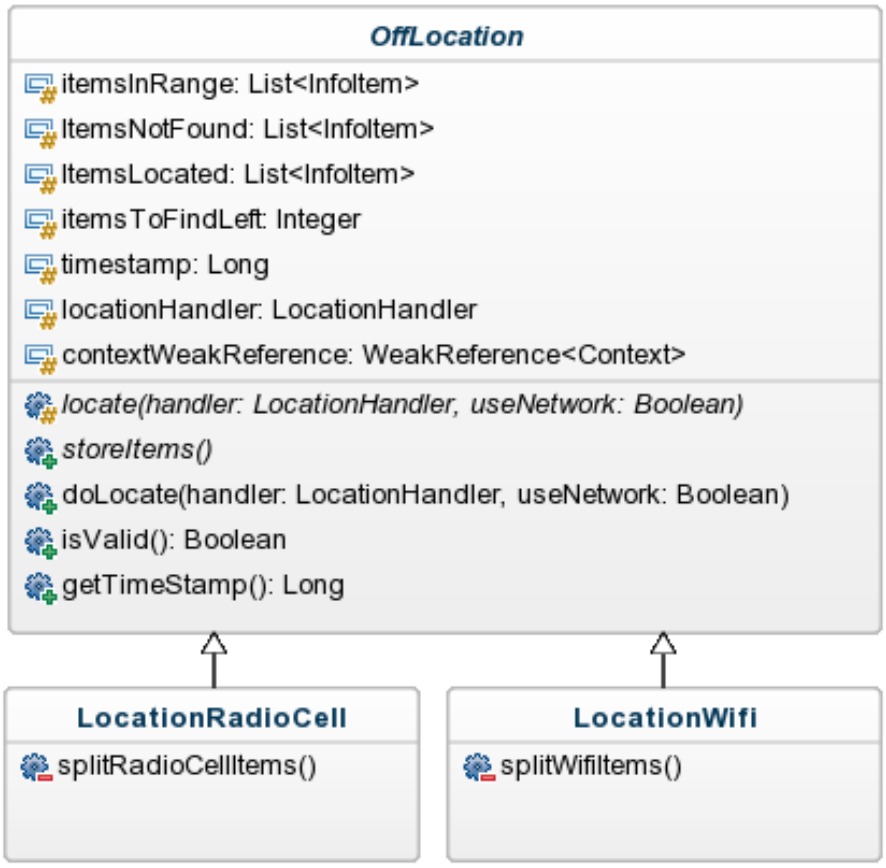
5.3 Trieda OffLocation

Táto trieda slúži ako základ pre každý typ lokalizácie implementovaný knižnicou a je definovaná ako abstraktná trieda. Týmto spôsobom sú určité základné vlastnosti, členské premenné a funkcionalita spoločné pre všetkých potomkov tejto triedy a je možné s nimi pracovať rovnako. Zároveň poskytuje individuálne spracovanie určitých vlastností prostredníctvom deklarovaných abstaktných metód, ktoré je potrebné implementvať až v triedach, ktoré sú od nej odvodené. V Zhrnutí každý druh lokalizácie v implementovanej knižnici je potomkom triedy `OffLocation` a implementuje abstraktné metódy v závislosti od vlastnej špecifickosti.

Táto trieda obsahuje niekoľko dôležitých členských premenných s prístupom protected, ktoré sú teda viditeľné aj pre potomkoch triedy a priamo k ním pristupujú. Skupinu základňových staníc alebo prístupových bodoch WIFI reprezentovaných triedami ktoré implementujú rozhranie `InfoItem` ukladá trieda v troch objektoch typu `List<InfoItem>`:

- `List<InfoItem> itemsInRange` Obsahuje objekty ktoré sa nachádzajú aktuálne v okolí zariadenia. Objekty z tohto zoznamu sú neskôr prerozdelené do ďalších dvoch zoznamov počas lokalizovania zariadenia.
- `List<InfoItem> itemsNotFound` V tomto zozname sa nachádzajú objekty ktoré sa nepodarilo nájsť ani v cache databáze a ani na API ktoré knižnica používa na sťahovanie informácií o staniciach.
- `List<InfoItem> itemsLocated` V tomto zozname sú zaradené objekty ktorých poloha je už známa a môže sa s nimi pracovať pri určovaní polohy.

Inicializácia týchto zoznamov prebieha v konštruktore. Ďalšou členskou premennou je objekt typu `LocationHandler`, ktorý je využitý neskôr po určení polohy a je popísaný v kapitole 5.4. Premenná `long timeStamp` je, ako jej názov vypovedá sám, použitá ako časová známka. Slúži na odlíšenie viacerých rôznych instancií triedy 5.3 a jej potomkov z časového hľadiska a určenie, či vypočítaná poloha je ešte aktuálna. Poslednou premennou je `int itemsToFindLeft`. Táto premenná sa používa v prípade že zaznamenané BTS alebo WIFI body v okolí nie sú ešte uložené v cache knižnice a je potrebné informácie o ich polohe stiahnúť z API. Na túto úlohu sa používa knižnica `Volley`, ktorá spôsobuje v niektorých prípadoch situáciu kedy príde odpoveď z API viac krát po sebe a preto parameter `int itemsToFindLeft` informuje o tom, či je ešte potrebné počkať na odpoveď z API alebo už neostala žiadna BTS stanica alebo WIFI bod ktoré nie sú lokalizované. Situácia kedy prišla odpoveď s polohou staníc dva krát bla veľmi zriedkavá,



Obr. 9: Diagram tried použitia triedy OffLocation.

no bolo nutné ju ošetriť pretože spôsobovala problémy. Priamými potomkami tejto triedy sú triedy `LocationRadioCells` a `LocationWifi` ako je vidieť na diagrame tried na obrázku 9.

Súčasťou implementujúcich tried sú aj vnútorné triedy, ktoré implementujú určité rozhrania, no v diagrame nie sú znázornené z dôvodu robustnosti a rozvetvenosti tejto štruktúry, ktorá je popisovaná v kapitole 5.7. Ako vidieť z diagramu, obe tieto triedy implementujú dve abstraktné metódy:

- `void locate(LocationHandler handler, boolean useNetwork)` - má za úlohu lokalizovať polohu všetkých zaznamenaných BTS staníc alebo WIFI bodov. V tejto metóde prebieha už spomínané prerozdelenie objektov reprezentujúcich tieto okolité stanice a body do zoznamov podľa toho či informácie o ich polhe už sú v cache alebo nie a prípadné sťahovanie informácií o ich polohe z API. Rozhranie `LocationHandler` slúži na notifikovanie o výsledkoch pomocou jeho calback metód. Parameter `boolean useNetwork` umožňuje

vynútiť pracovať iba s informáciami ktoré sú v cache databáze a v prípade že je potrebné nejaké ich stiahnúť, je tento krok preskočený a s týmito objektami sa nepracuje.

- `void storeItems()` - ako už jej samotný názov napovedá, slúži na vytvorenie záznamov o BTS alebo WIFI v internej SQLite databáze a ich uloženie. Na ukladanie informácií o BTS súlžia triedy `CellsDataSource` a `CellsSqliteHelper` a na ukladanie a spracovávanie informácií o prístupových bodoch WIFI slúžia triedy `WifiDataSource` a `WifiSQLiteHelper`.

5.4 Rozhranie LocationHandler

Toto rozhranie slúži ako callback handler prenotifikovanie o úspešnom alebo neúspešnom zistení polohy zaznamenaných okolitých BTS staníc alebo prístupových bodov WIFI. Rozhranie obsahuje deklarované dve abstraktné metódy s prázdným návratovým typom void:

- `onLocationSuccess(List<InfoItem> items, LocationType type, long timeStamp)` - slúži na notifikovanie v prípade že sú okolité stanice alebo prístupové body úspešne lokalizované a je možné pracovať ďalej pri určení polohy mobilného zariadenia. Vstupný parameter `void locatedItems` je zoznam týchto staníc s ktorými sa ďalej pracuje pri určovaní polohy. Parameter `LocationType type` je vytvorený enum s hodnotami `Wifi` alebo `RadioCell` pre potrebu odlíšenia, či ide o lokalizáciu pomocou základňových staníc alebo prístupových bodov WIFI. Parameter `timeStamp` predstavuje časovú známku spomínanú v kapitole 5.3

- `onLocationError(String error, long timeStamp)` - slúži na notifikovanie v prípade problému pri pokuse o lokalizovanie jednotlivých staníc alebo prístupových bodov. Parameter `error` reprezentuje chybovú správu pre upresnenie príčiny chyby a parameter `timeStamp` predstavuje časovú známku.

5.5 Priority v lokalizovaní

Určená poloha pomocou BTS staníc sa dosť líši od polohy určenej pomocou WIFI bodov v okolí a z tohto dôvodu bolo potrebné priradiť týmto dvom typom lokalizácie ich priority. Lokalizovanie pomocou BTS môže byť nepresné aj v husto osídlenej oblasti s dostatočným počtom staníc, no v prípade lokalizácie pomocou WIFI bodov tento problém nie je, hlavne kvôli ich menšiemu dosahu. V prípade dostupnosti oboch typov informácií - BTS aj WIFI bodov, je v priebehu výberu ďalšieho kroku uprednostnená lokalizácia pomocou WIFI. Dôvodom je už spomínaná nepresnosť lokalizácie pomocou BTS a toto rozhodnutie potvrdzujú aj namerané výsledky prezentované v kapitole 7. Knižnica umožňuje aj vynútiť konkrétny typ lokalizáce a priority sú v tom prípade ignorované.

5.6 Cache dát pomocou SQLite

Dáta o polohe BTS staníc a prístupových bodoch WIFI sú ukladané pomocou SQLite databázy. Z tejto databázy sú dáta používané v prípade, ak prvky v okolí mobilného zariadenia sú lokalizované a ich poloha je v databáze uložená. V tomto prípade nie je potrebné vykonávať HTTP requesty informácie o polohe vypočítať z už uložených dát. Všeobecným vzorom pre vytvorenie databázy je dvojica tried. Jedna trieda slúži ako helper, teda pomocná trieda a dedí z triedy `SQLiteOpenHelper` a ma za úlohu vytváranie a správu databázy. Druhá trieda obsahuje metódy na spracovávanie dát ukladanýchv databáze.

5.6.1 Databáza BTS staníc

Pre ukladanie dát o BTS staniciach slúžia triedy `CellSQLiteHelper` ako spomínaný helper na správu databázy a `CellsDataSource` obsahujúci metódy na prácu s databázou. Záznam jednej BTS stanice obsahuje nasledovné stĺpce:

- `COLUMN_CELL_ID` - ID základňovej stanice použité ako unikátny primárny klúč
- `COLUMN_TYPE` - popisuje typ generácie ku ktorej stanica patrí - GSM, WCDMA, CDMA, LTE
- `COLUMN_MNC` - Mobile Network Code
- `COLUMN_MCC` - Mobile Country Code
- `COLUMN_LAC` - Locaton Area Code
- `COLUMN_NETWORK_ID` - ID siete používané pre CDMA stanice namiesto Location Area Code
- `COLUMN_SYSTEM_ID` ID systému používané pre CDMA stanice namiesto Mobile Network Code
- `COLUMN_LAT` - Latitude parameter zemepisnej šírky polohy stanice
- `COLUMN_LON` - Longitude parameter zemepisnej dĺžky polohy stanice
- `COLUMN_TAC` - Tracking Area Code používaní pri LTE staniciach namiesto Location Area Code
- `COLUMN_LOCATED` - informácia o tom či je stanica lokalizovaná a jej poloha je známa alebo je potrebné polohu zistiť

5.6.2 Databáza prístupových bodov WIFI

Pre ukladanie dát o prístupových bodoch WIFI slúžia triedy `WifiSQLiteHelper` ako helper na vytváranie a správu databázy a `WifiDataSource` obsahujúci metódy na prácu s databázou. Záznam jedného prístupového bodu WIFI obsahuje nasledovné stĺpce:

- `COLUMN_BSSID_LONG` - ID prístupového bodu použité ako unikátny primárny kľúč záznamu. ID je vytvorené z adresy BSSID, ktorá je parsovaná na číslo typu `long` metódou `long parseLong(String string, int radix)`, ktorá je statická a patrí pod triedu triedy `Long`.
- `COLUMN_BSSID` - adresa prístupového bodu.
- `COLUMN_SSID` - názov siete prístupového bodu.
- `COLUMN_MAC` - MAC adresa rozhrania prístupového bodu
- `COLUMN_LOCATED` - informácia o tom či je prístupová bod lokalizovaný a jeho poloha je známa alebo je potrebné polohu zistiť
- `COLUMN_LAT` - Latitude parameter zemepisnej šírky polohy stanice
- `COLUMN_LON` - Longitude parameter zemepisnej dĺžky polohy stanice

5.7 Získavanie polohy BTS a WIFI pomocou HTTP

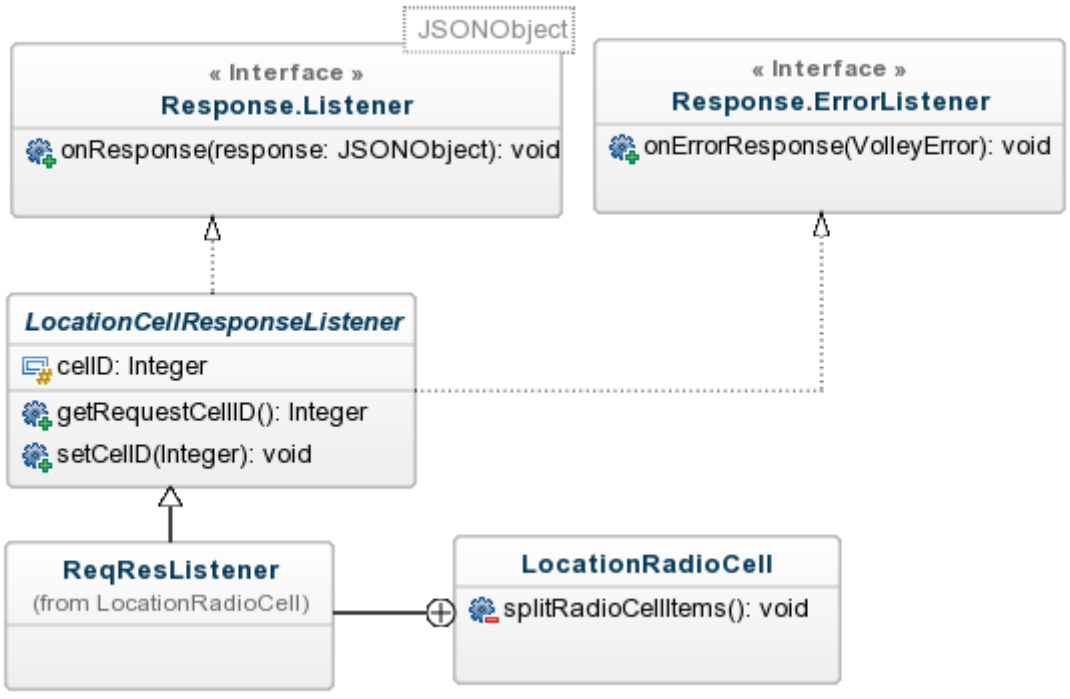
V prípade že informácie o polohe okolitých BTS alebo prístupových bodoch WIFI nie sú k dispozícii z cache SQLite databázy, je potrebné tieto informácie stiahnúť zo servera. Informácie sú získavané prostredníctvom HTTP requestov, ktoré sú vytvorené pomocou knižnice `Volley`. Informácie o polohe jednotlivých BTS staníc alebo prístupových bodoch nie sú oficiálne dostupné. Z tohto dôvodu je dostupných niekoľko služieb, ktoré majú vybudovanú databázu na základe meraní ktoré sú odosielané užívateľmi na ich servery. Ide teda o približnú polohu týchto staníc a nie je úplne presná. Medzi známe patria napríklad web OpenCellid, OpenSignal, OpenWifi služby combain positioning solutions alebo Mozilla Location Service, no v prípade ich použitia sa vyskytujú komplikované obmedzenia. Napríklad vynútená potreba odosielania nameraných dát v pomere na počet requestov, prípadne mesačná platba za používanie týchto API. Z tohto dôvodu, vzhľadom na snahu o čo najjednoduchšie vytvorenie knižnice, ktorá by neomedzovala používateľa, je použté Google Maps Geolocation API. Na používanie tohto API je potrebné v GoogleDeveloper Console vytvoriť a pomenovať vlastný projekt a v sekcii Credentials vygenerovať API kľúč, ktorý je potrebný odosielať s každým requestom na toto API [9, 10]. HTTP requesty, ktoré sú psoielané na API používajú POST metódu s nasledujúcou URL:

https://www.googleapis.com/geolocation/v1/geolocate?key=API_KEY

Parameter API_KEY predstavuje vygenerovaný API kľúč. Request a response od API sú formátované ako JSON. Na zachytávanie odpovedí na tieto requesty sú vytvorené abstraktné triedy implementujúce rozhrania `Response.Listener<JSONObject>` a `Response.ErrorListener`. Pomenované sú `LocationCellResponseListener` a `LocationWifiResponseListener` a sú implemenované vo vnútri už spomínaných tried `LocationRadioCells` a `LocationWifi` ako vnorené privátne triedy a sú používané iba na tomto mieste. Vzhľadom na to že ide o abstraktné triedy, ktoré implementujú dve rozhrania, nie sú metódy z týchto rozhraní implementované ani v abstraktných triedach, ale až vo vnorených triedach ktoré z nich dedia. Metódy na spracovávanie callbackov po vytvorení requestov ktoré je treba implementovať sú nasledovné:

- `void onResponse(JSONObject response)` informuje o úspešne spracovanom requeste a obsahuje ako parameter JSON objekt, ktorý obsahuje informácie o polohe BTS stanice alebo prístupového bodu WIFI.
- `void onErrorResponse(VolleyError error)` informuje o neúspešnom requeste a pomocou parametra `error` je možné zistiť dôvod zlyhania a reagovať naň.

Táto štruktúra znázornená pomocou diagramu tried na obrázku 10, konkrétne pre triedu `LocationRadioCell`:



Obr. 10: Diagram tried pre triedy LocationCellResponseListener a LocationRadioCell.

Na výpise kódu4 je pre ucelenie obrazu ukážka implementácie popisovanej abstraktnej triedy ako vnorenej pre triedu `LocationRadioCells`. Implementácia pre triedu `LocationWIFI` je obdobná a nie je znázornená pre zbytočnú duplicitu. V ukážke sa tiež nenachádza telo metód a ani obsah triedy `LocationRadioCells` pre jednoduchšiu čitateľnosť.

```
public class LocationRadioCells extends OffLocation {

    private final class ReqResListener extends LocationCellResponseListener{

        @Override
        public void onResponse(JSONObject response) {
        }

        @Override
        public void onErrorResponse(VolleyError error) {
        }
    }
}
```

Výpis 4: Kostra triedy LocationRadioCells

5.7.1 JSON request pre získavanie polohy BTS

Pri tvorbe requestu obsahuje telo vo formáte JSON nasledovné dva parametre. Prvým je `boolean` hodnota uložená pod kľúčom `considerIp` nastavená na `false`, pretože geolokalizačná služba môže použiť namiesto základňovej stanice IP adresu requestu. To má za výsledok veľmi nepresú polohu, a deje sa v prípade ak sa BTS stanicu nepodarilo lokalizovať. Ak je hodnota considerIp nastavená na `false` a odpoveď od API je kód `404`, znamená to že požadovaná BTS stanica nemohla byť lokalizovaná. S takouto stanicou sa ďalej pri určovaní polohy nepracuje. Druhým parametrom requestu je pole, v ktorom sú posielané objekty BTS staníc. V tomto prípade obsahuje len jeden objekt, pretože je potrebné zistiť polohu len jednej stanice. Ak by sa v poli nachádzalo viac objektov, výsledkom requestu by bola približne vypočítaná poloha na strane API, čo v tomto prípade nie je žiadané a preto sa pre zisťovanie polohy stanice posiela v poli vždy len jeden objekt. Toto pole je uložené pod kľúčom `cellTowers`. Objekt BTS stanice vo formáte JSON obsahuje nasledovné kľúče:

- `cellId` - ID základňovej stanice.
- `locationAreaCode` - Location Area Code pre GSM a WCDMA siete alebo Network ID pre CDMA siete.

- `mobileCountryCode` - Mobile Country Code stanice
- `mobileNetworkCode` - Mobile Network Code pre GSM a WCDMA siete alebo System ID pre CDMA siete.

Telo môže obsahovať aj ďalšie nepovinné parametre, ktoré sa v tomto prípade nepoužívajú. Vo výpise kódu 5 je ukážka tela JSON requestu s vyplnenými dátami požadovanej BTS.

```
{
    "considerIp": false,
    "cellTowers": [
        {
            "cellId": 42,
            "locationAreaCode": 415,
            "mobileCountryCode": 310,
            "mobileNetworkCode": 410
        }
    ]
}
```

Výpis 5: Telo JSON requestu s dátami o BTS

5.7.2 JSON request pre získavanie polohy prístupového bodu WIFI

Telo requestu pre WIFI je podobné ako telo requestu pre BTS. Súčasťou je rovnako ako v predchádzajúcom prípade kľúč - `considerIp` s hodnotou `false`. Druhý parameter je tiež pole, no je uložené pod kľúčom `wifiAccessPoints`. V tomto prípade je ďalšia odlišnosť od predchádzajúceho a tou je, že v poli sa už nenachádza len jeden objekt predstvujúci WIFI objekt, ale dva. Táto podmienka je definovaná v použitom API a komplikuje získavanie polohy prístupového-bodu v prípade že je k dispozícii len jeden prístupový bod [10]. Pri tvorbe requestov sa preto musia vytvoriť dvojice zo získaných objektov WIFI bodov zaznamenaných v dosahu mobilného zariadenia. Vyberajú sa vždy dva objekty v priamom slede za sebou počas prechádzania zoznamu v ktorom sú uložené. Pre každý objekt na indexe i je vybraný objekt na indexe `i+1`. Ak je množstvo n objektov v zozname väčší ako 2, je pre posledný objekt vybraný do páru zo zoznamu náhodne zvolený objekt v rozsahu `n-1`.

Objekt prístupového bodu WIFI vo formáte JSON obsahuje nasledovné kľúče:

- `macAddress` - MAC adresa pístupového boduslúžiaca ako identifikátor
- `signalStrength` - sila signálu v jednotkách dBm

Rovnako ako v predchádzajúcom prípade je možné doplniť ďalšie nepovinné parametre. Vo výpise kódu 6 je ukážka tela JSON requestu s vyplnenými dátami požadovaných prístupových bodov WIFI.

```
{
  "considerIp": false,
  "wifiAccessPoints": [
  {
      "macAddress": "01:23:45:67:89:AB",
      "signalStrength": 8
  },
  {
      "macAddress": "01:23:45:67:89:AC",
      "signalStrength": 4
  }
}
```

Výpis 6: Telo JSON requestu s dátami o prístupovom bode WIFI

5.7.3 JSON response pre prístupového bodu WIFI a BTS

Štruktúra získaného JSON objektu, v ktorom sa nachádzajú informácie o polohe konkrétneho prvku základňovej stanice alebo prístupového bodu WIFI je pre obe verzie rovnaká a je znázornená na výpise 7

```
{
    "location":
    {
        "lat":49.4764931,
        "lng":18.7913665
    },
    "accuracy":2413
}
```

Výpis 7: Telo JSON response s dátami o BTS alebo WIFI

Hodnoty uložené pod kľúčami `lat` a `lng` sú súradnice hľadaného objektu a používajú saďalej pri výpočte približnej polohy moblného zariadenia, hodnota pod kľčom `accuracy` predstavuje približnú presnosť súradníc, no tento parameter sa nevyužíva, pretože výpočet polohy mobilného zariadenia je úlohou implementovanej knižnice na základe úrovni príjmaných signálov.

5.8 Výpočet polohy zariadenia

Najdôležitejím krokom po získani polohy jednotlivých staníc v okolí je určenie hrubej polohy mobilného zariadenia. Tento výpočet má na starosti trieda `LocationCalculator`, ktorá je koncipovaná pomocou návrhového vzoru `SINGLETON` pre potrebu existencie iba jedinej instancie s informáciami o výsledku vypočítanej polohy a jednoducému prístupu k ním s istotou že daný výsledok je aktuálny.

Samotný výsledok je reprezenovaný vnorenou triedou `Result` a jej instancia je súčasťou triedy `LocationCalculator`. Táto instancia je definovaná ako `final` a preto musí byť vytvorená v konštruktore triedy `LocationCalculator`. Počas výpočtu polohy mobilného zariadenia sa danej instancii pomocou verejných metód nastavia vždy aktálne vlastnosti na základe výpočtov. Obsauje všetky potrebné informácie nei len o vypočítanej polohe, ale aj o tom aký počet BTS staníc alebo WIFI bodov bol použitý na výpočet a podobne. Tieto informácie je možné získať z akéhokoľvek miesta volaním `LocationCalculator.getInstance().getLocationResult()`. Po takomto volaní je k dispozícii vždy len jeden a aktuálny výsledok určovania polohy a poskytuje informácie na spracovanie podľa potrebnej situácie.

Výpočet začína volaním metódy `void calculateLocation(List<InfoItem> itemsInRange, LocationHandler.LocationType type)`. Úlohou tejto metódy je na základe vstupného parametra `type` určiť typ lokalizácie a na jeho základe volaním už privátných metód spustiť samotný výpočet, ktorý pracuje s prvkami vstupného parametru `itemsInRange`. Výpočet polohy mobilného zariadenia prebieha v jednej z dvoch metód:

- `void calculateWifiLocation(List<InfoItem> items)` pre lokalizáciu pomocou informácií o prístupových bodoch WIFI
- `void calculateCellLocation(List< InfoItem> cellsInRange)` pre lokalizáciu pomocou informácií o staniciach BTS

Priebeh výpočtu je v oboch metódach rovnaký, líšia sa len v úvodnom spracovaní dát. Dôvodom rovnakého výpočtu v prípade oboch typov je nemožnosť zistiť vyžarovací výkon týchto zariadení a z toho plynie nemožnosť zistiť približné pokrytie územia signálom. Kvôli tomu sa pracuje s každým objektom v rámci typu lokalizácie rovnako - objekty sú navzájom rovnocenné z pohľadu výkonu. Kvôli zjednodušeniu predstavy sa dá povedať že poloha mobilného zariadenia sa určí ako priesečník kružníc, ktorých stredom je poloha jednotlivých staníc s polomerom relatívnym k úrovni prijatého signálu od danej stanice. Úroveň signálu, ktorá predstavuje veľkosť okruhu pokrytého signálom a má určenú hranicu ktorá predstavuje maximálny polomer. Tento maxiálny polomer je určený minimálnou úrovňou prijatého signálu, ktorá sa líši vzhľadom na typ technológie, no pracuje sa s ním ako s kladným číslom. Dôležitou súčasťou je aj stanovený rozsah, ktorý je pri výpočte polohy použitý na stanovenie váhy pre jednotlivé úrovne signálov vzhľadom na úrovne signálov ostatných prvkov.

Samotný výpočet polohy zariadenia je rozdelený na niekoľko krokov. V prvom rade je potrebné súradniciam každého prvku BTS alebo prístupovému bodu WIFI priradiť na základe ich úrovne prijatého signálu váhu v cykle ktorý prejde všetky vstupné prvky. Váha pre jeden prvok sa vypočíta nasledovne: $(\text{min\_level} + \text{real\_level})/\text{range}$.

Premenná `min_level` predstavuje spomínanú minimálnú úroveň prijatého signálu ako kladné číslo, `real_level` predstavuje skutočnú hodnotu úrovne prijatého signálu a `range` stanovený rozsah pre konkrétnu technológiu. Hodnoty jednotlivých rozsahov sú znázornené v tabuľke 2.

typ	minimum	rozsah
GSM / WCDMA	114	113
CDMA	106	105
LTE	141	140
WIFI	100	45

Tabuľka 2: Minimálne použiteľné úrovne signálov pre jednotlivé technológie

Na príklade 1 je ukážka takéhoto výpočtu. Počas výpočtu jednotlivých váh sa zároveň spočítava aj ich suma, ktorá je neskôr použitá pri aplikovaní vypočítaných váh na výslednú polohu.

Príklad 1
Máme objekt `InfoItem` ktorý reprezentuje jednu BTS stanicu pre GSM sieť. Úroveň prijatého signálu od tejto stanice je -70 dBm. Keďže ide o GSM stanicu, minimálna úroveň použitá vo výpočte podľa tabuľky 2 je `114` a rozsah `113`. Po dosadení do vzorca dostaneme $(114 + (-70))/113$, čo nám dá po zaokrúhlení na stotiny výsledok `0,39` (pri výpočte v kóde sa hodnota nezaokrúhľuje, pracuje sa so surovým, čo najjemnejším výsledkom). Takto vypočítana váha sa priradí k objektu a pokrauje sa výpočtom ďalšej váhy pre ďalší objekt.

■

Po určení váhy jednotlivých signálov sa pristúpi k výpočtu polohy. Opäť sa prechádzajú všetky prvky v cykle a pomocou už vypočítaných váh na základe úrovni ich signálov. V tomto cykle sa počítajú výsledné latitude a longitude polohy mobilného zariadenia samostatne. Pre výpočet `latitude` aj `longitude` súradnice polohy je vzorec rovnaký a môže byť vyjadrený nasledujúcim pseudovzorcom:

$$\text{final\_coordinate} = \text{item\_coordinate} * (\text{signal\_weight}/\text{signal\_sum})$$

Takto získaný dielčí výsledok sa pripočíta k výslednej polohe, ktorá je tvorená súčtom polôh jednotlivých prvkov. Parameter `item_coordinate` je latitude alebo longitude aktuálneho prvku, `signal_weight` je vypočítaná váha daného prvku a `signal_sum` súčet všetkých váh aktuálne používaných prvkov. Výsledná poloha mobilného zariadenia je teda vypočítana pomocou váženého priemeru - súčet dielčích súčinov jednotlivých súradníc polohy stanice a jej váhy vypočítanej na základe úrovne prijatého signálu. Po dokončení cyklu je poloha zariadenia vypočítaná a je dostupná pre ďalšie spracovanie.

5.8.1 Určenie presnosti

Určená poloha pomocou knižnice vo svojej podstate nie je absolútne presná a preto je potrebné určiť aj rádius, v ktorom by sa mala skutočná poloha nachádzať. Pôvodne som rádius určil pomocou vypočítaných váh na základe úrovne prijatého signálu pre určenie polohy, no v určitých prípadoch počas testovania bola skutočná GPS poloha mimo aj tohto rádiusu. Z tohto dôvodu som zvolil rádius ako vzdialenosť medzi určenou polohou a najvzdialenejšou zo základňových staníc alebo prístupovým bodom WIFI, pomocou ktorých je poloha počítaná. Rádius je teda určený ako najhoršia možná varianta.

5.9 Trieda OfflineLocationSource

Súčasťou knižnice OfflineLocation je aj jej vlastný poskytovateľ polohy, ktorý slúži ako poskytovateľ polohy mobilného zariadenia. Podmienkou je implementácia rozhrania LocationSource. To umožňuje použiť namiesto vstavaného poskytovateľa polohy pre objekt GoogleMap vlastný, upravený tak, aby poskytoval polohu ktorá je získavaná pomocou samotnej knižnice OfflineLocation a jej metód. Použitím tohto rozhrania je potrebné implementovať dve metódy s návratovým typom `void` :

- `activate (LocationSource.OnLocationChangedListener listener)` - úlohou tejto metódy je aktivovať vlastného providera, nastaviť potrebné parametre a notifikovať zaregistrovaný `listener` ktorý je predávaný prostredníctvom parametra metódy. V tejto metóde je nastavený typ lokalizácie a vytvorený objekt vnorenej privátnej triedy - `LocationUpdater` ktorý má za úlohu periodicky získavať polohu zariadenia.
- `deactivate()` - slúži na deaktiváciu vlastného providera, ukončí notifikovanie zaregistrovaného listenera pomocou callback metód a ukončí periodické získavanie polohy zariadenia.

V tejto triede je nadefinovaný typ lokalizácie, ktorá je aktuálne využívaná pomocou kľúčového slova enum, je pomenovaný `LocationType` a delí sa na tri druhy:

- `Cell` - vyhľadáva a zisťuje iba lokalizáciu pomocou rádiových sietí
- `Wifi` - vyhľadáva a zisťuje iba lokalizáciu pomocou prístupových bodov WIFI sietí
- `Combined` - vyhľadáva a zisťuje lokalizáciu oboch typov - pomocou rádiových sietí a pomocou prístupových bodov WIFI sietí. V tomto prípade, ak sú k dispozíciidáta oboch typov sa rozhoduje na základe priorít, ktoré boli spomínané v kapitole 5.5

Defaultne je ako typ nastavený typ `Combined`, no je možné ho zmeniť pomocou metódy `setLocationType(OfflineLocationSource.LocationType type)`. Ďalšou podstatnou časťou tejto triedy je frekvencia získavania polohy mobilného zariadenia. Táto frekvencia je definovaná ako pauza v milisekundách, po ktorú je vlákno uspané do ďalšieho zisťovania polohy zariadenia.

K dispozícii sú preddefinované 4 konštanty typu `long`, ktoré reprezentujú zvolený čas uspania vlákna a sú znázornené v tabuľke 3.

Názov konštanty	hodnota
UPDATE_FREQUENCY_LOW	30000
UPDATE_FREQUENCY_MEDIUM	20000
UPDATE_FREQUENCY_HIGH	5000
UPDATE_FREQUENCY_MAX	2500

Tabuľka 3: Hodnoty konštánt pauzy vlákna v milisekundách

5.9.1 Vnorená trieda LocationUpdater

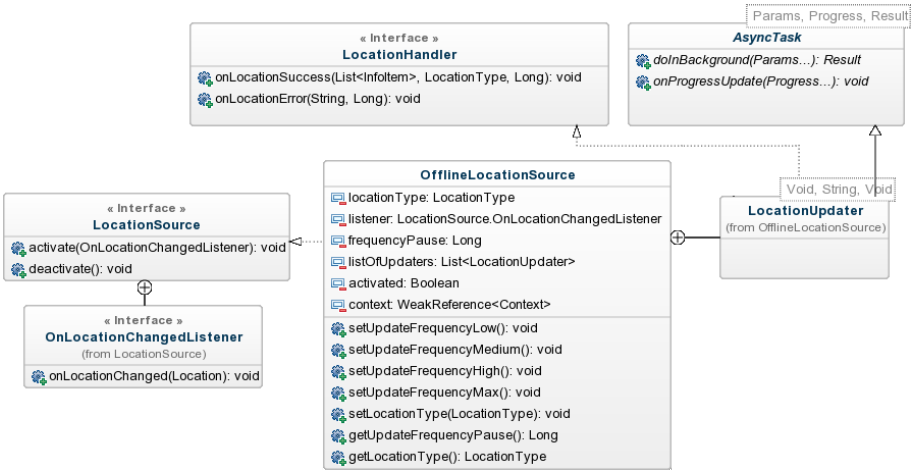
Získavanie polohy zariadenia prebieha periodicky v cykle na pozadí pomocou vnorenej triedy `LocationUpdater`, ktorá dedí z triedy `AsyncTask` a implementuje rozhranie `LocationHandler`, pomocou ktorého callback metód notifikuje vonkajšiu triedu o úspechu, prípadne neúspechu získania polohy mobilného zariadenia. Táto trieda je súčasťou triedy `OfflineLocationSource` ako privátna a vnorená trieda. Pri vytvorení tejto triedy je nutné zadefinovať tri generické typy všeobecne nazývané ako[14]:

- `Params` - vstupné parametre metódy `doInBackground(Params...)`
- `Progress` - vstupné parametre metódy `onProgressUpdate(Progress...)`
- `Result` - vstupné parametre metódy `onPostExecute(Result...)`

V tomto prípade sa používa iba typ `Progress`, ktorý je definovaný ako `String`. Tento parameter je predávaný ako vstupný metóde `onProgressUpdate(String... values)` a slúži ako informácia o type použitej lokalizácie - buď pomocou rádiových sietí alebo pomocou prístupových bodov WIFI. Typy `Params` a `Progress` sú definované ako `Void`, čo znamená že metódy ktoré ich používajú nemajú žiadny vstupný parameter.

V metóde `doInBackground(Void... params)` prebieha v cykle skenovanie okolia mobilného zariadenia a jeho lokalizovanie, ktorého úspech alebo neúspech je zaznamenaný pomocou implementivaných callback metód `(List<InfoItem> items, LocationType type, long timeStamp)` a `onLocationError(String error, long timeStamp)`. Po určení polohy mobilného zariadenia sa metódou `onProgressUpdate(String... values)` notifikuje vonkajšia trieda `OfflineLocationSource`. Notifikovanie prebieha prostredníctvom listenera, ktorý slúži na zachytávanie informácií o zmene aktuálnej polohy a je zaregistrovaný pri spustení získavania polohy zavolaním metódy `activate (LocationSource.OnLocationChangedListener listener)`. Po tejto iterácii cyklu je vlákno uspané na určitý počet milisekúnd, ktoré sú nastavené podľa už popisovanej tabuľky 3.

Ukážka štruktúry tejto triedy spolu s jej vonkajšou triedou je zobrazená v diagrame tried na obrázku 11. V diagrame sa nenachádza celý obsah abstraktnej triedy AsyncTask kvôli čitateľnosti, pretože trieda patrí priamo do Android frameworku a je značne rozsiahla.



Obr. 11: Diagram tried pre triedu OfflineLocationSource.

5.10 Doplnková knižnica Volley

Pri implementácii knižnice bolo potrebné použiť aj dodatočnú externú knižnicu s názvom Volley. Jej úlohou je vytváranie a spracovavánie HTTP requestov, ktoré sú v prípade implementovanej lokalizačnej knižnice potrebné na sťahovanie informácii o polohe základňových staníc alebo prístupových bodov WIFI. Knižnica ponúka nasledujúcu funkcionalitu:

- automatické rozvrhovanie requestov
- viacero súbežných sieťových pripojení
- transparentné cache-ovanie response dát na disku, prípadne v pamäti v konzistencii so štandardným HTTP cache-ovaním
- nastavovanie priorít jednotlivých requestom
- poskytuje API k rušeniu requestov
- umožňuje jednoduché prispôsobenie tvorby requestov a logiky vzhľadom na potreby aplikácie
- spoľahlivé usporiadanie, ktoré umožňuje pracovať s UI aj v prípade asynchronne spustených operácii
- nástroje pre debugovanie a sledovanie operácii

Knižnica je vhodná na prácu s RPC operáciami, umožnuje jednoduchú integráciu s akýmkoľvek protokolom a podporuje spracovávanie RAW reťazcov, obrázkov a formátu JSON. Naopak nie je vhodná pre spracovávanie veľkého množstva dát, ako napríklad streamovanie. Použitie tejto knižnice preto v prípade tejto diplomovej práce úplne vyhovuje, vzhľadom na malé množstvo prenášaných dát [15].

5.10.1 Používanie knižnice Volley

Pre prácu s knižnicou je potrebné vytvoriť instanciu triedy `RequestQueue`, ktorej sú pridávane instancie triedy `Request`. Prostredníctvom insancie `RequestQueue` sú spravované sieťové operácie, čítanie a zápis cache dát a spracovávanie RAW response dát. Na základe tohto prístupu je v implementovanej lokalizačnej knižnici vytvorená trieda `RequestExecutor` obstaravajúca tvorbu týchto objektov a prácu s nimi. Na implementáciu tejto triedy je použitý návrhový vzor `SINGLETON` ktorý zabezpečuje existenciu iba jednej instancie tejto triedy. Funkcionalita tejto triedy je tvorená dvoma verejnými metódami, ktoré umožňujú vytvoriť a vzkonávať requesty pre získavanie dát o BTS staniciach a prístupových bodoch WIFI:

- `void createAndExecuteCellRequest(...)` metóda slúži na vytváranie requestov pre získavanie dát o BTS staniciach. Vstupné parametre metódy sú objekty typu `Context` nutný k získaniu instancie triedy `RequestQueue`, `CellInfoItem` potrebný na vytvorenie requestu a nastavenie potrebných dát o hľadanej stanici a `LocationCellResponseListener` slúžiaci na notifikovanie o odpovedi.
- `void createAndExecuteRequest(...)` metóda slúži na vytváranie requestov pre získavanie dát o prístupových bodoch WIFI. Vstupné parametre metódy sú objekty typu `Context` nutný k získaniu instancie triedy `RequestQueue`, dva objekty `WifiInfoItem` kvôli použitiu Google Maps API potrebné na vytvorenie requestu a `LocationWifiResponseListener` slúžiaci na notifikovanie o odpovedi.

Trieda obsahuje ďalšie dve privátne metódy, ktoré slúžia na vytvorenie JSON tela prejednotlivé requesty, ktoré je následne spracované:

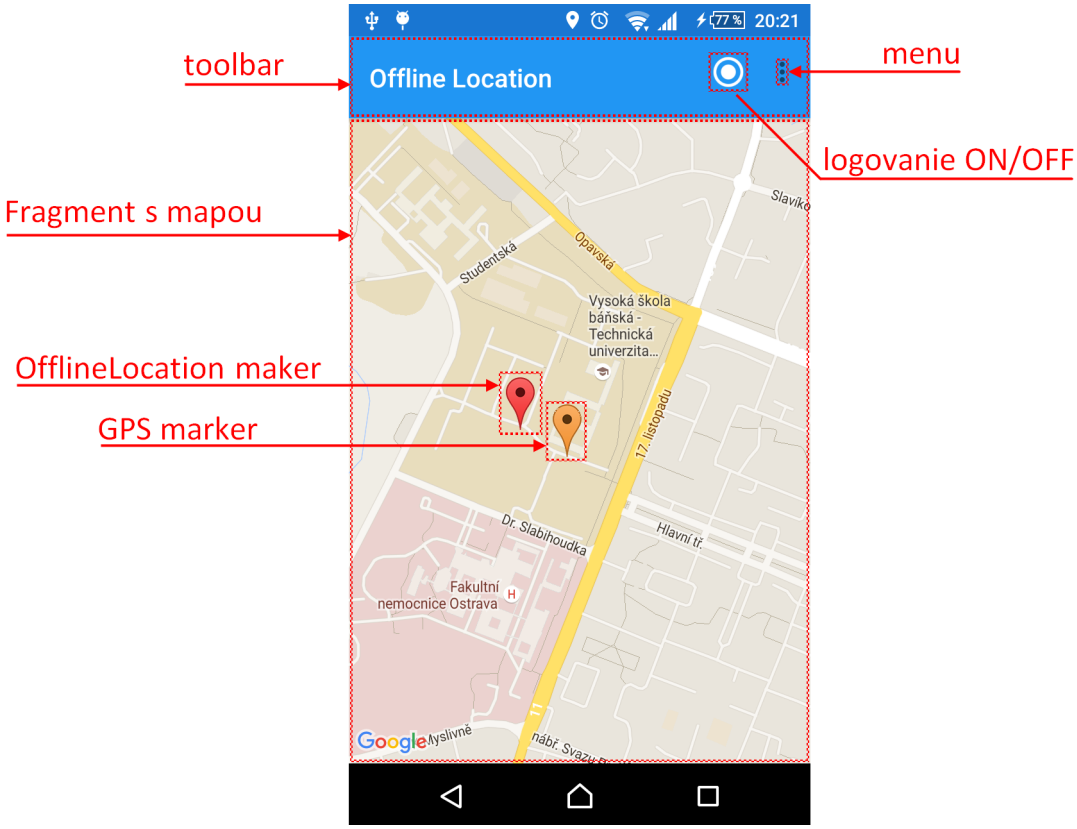
- `JSONObject createJSONCellRequestUrl(CellInfoItem item)` metóda
  má za úlohu vytvoriť JSON request pre získavanie dát o základňovej stanici
- `JSONObject createWlanRequestUrl(WifiInfoItem item, WifiInfoItem item2)` metóda
  má za úlohu vytvoriť JSON request pre získavanie dát o prístupových bodoch WIFI

6 Implementácia ukážkovej aplikácie

Ukážková aplikácia, ktorá demonštruje použitie vytvorenej knižnice je tvorená jedinou aktivitou, ktorá obsahuje mapu, na ktorej je zobrazená určená poloha mobilného zariadenia.

6.1 Užívateľské rozhranie

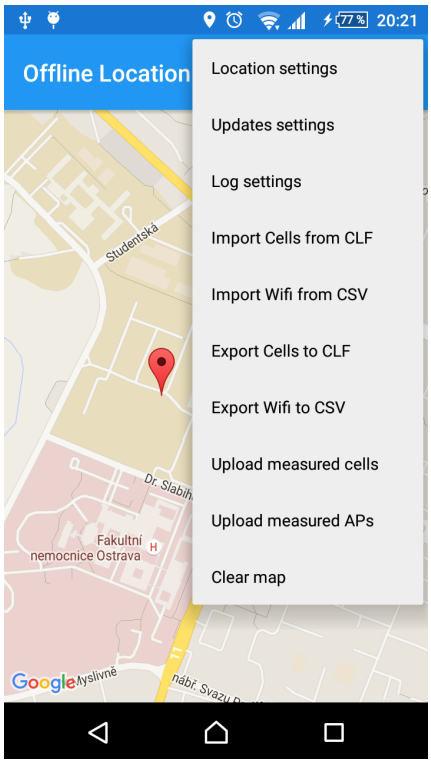
UI aplikácie je štandardne definované v xml súbore s názvom `activity_main.xml` v adresári `layout`. Dominantou aplikácie sú dva prvky - `Toolbar` element, ktorý obsahuje ovlácacie prvky aplikácie a mapa zobrazená pomocou elementu `Fragment` ktorý je staticky definovaný v spomínanom layoute. Možnosti nastavení aplikácie sú nadefinované ako `Menu` v súbore `menu_main.xml` v adresári `menu`. Okrem jednej položky ktorá je zobrazená ako samostatná akcia sú všetky definované ako skryté a zobrazia sa až po jeho rozbalení ako zoznam. Reakcia na výber každej položky je spracovaná v metóde `boolean onOptionsItemSelected(MenuItem item)`, ktorú implementuje hlavná aktivita aplikácie. Na obrázku 12 je ukážka vzhľadu aplikácie s popisom jednotlivých komponent.



Obr. 12: UI aplikácie - popis prvkov

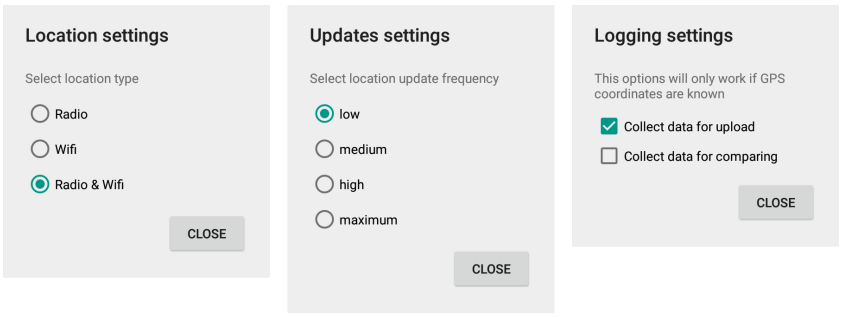
6.2 Možnosti aplikácie

Aplikácia poskytuj niekoľko nastavení a možností spracovania nameraných výsledkov pomocou menu ponuky, ktorá sa zobrazí po kliknutí na ikonu menu v `Tolbar` lište, ako je ukázané na obrázku 13:



Obr. 13: UI aplikácie - ukážka menu

V aplikácii sa nachádzajú aj tri dialogové okná, ktoré zobrazujú nastavenia pre niektoré možnosti z hlavného menu. Konkrétne sú to nastavenia typu lokalizácie, frekvencie aktualizácie a logovania dát. Ukážka vzhľadu a obsahu týchto okien sa nachádza na obrázku 14. Možnosť `Radio & Wifi` ponecháva rozhodnutie použiť typ lokalizácie automaticky, ako je popisované v kapitole 5.5.



Obr. 14: UI aplikácie - ukážka dialogových okien v aplikácii

Funkcie jednotlivých možností sú nasledovné:

- **Location setings** - umožňuje nastavenie typu lokalizácie (Radio - Wifi - Radio/Wifi)
- **Updtes settings** - umožňuje nastavenie frekvencie aktualizovania polohy
- **Log settings** - umožňuje nastavenie logovania dát
- **Import cells CLF** - spustí import základňových staníc z CLF súboru
- **Import WIFI from CSV** - spustí import WIFI AP z CSV súboru
- **Export cells to CLF** - spustí export základňových staníc do CLF súboru
- **Export WIFI to CSV** - spustí export WIFI AP do CSV súboru
- **Upload measured cells** - spustí upload základňových staníc z vybraného súboru
- **Upload measured APs** - spustí upload WIFI AP z vybraného súboru

  Clear map - umožňuje nastavenie typu lokalizácie (Radio - Wifi - Radio/Wifi)

6.3 Zobrazovanie polohy

Ukážková aplikácia umožňuje zbrazovať na mape polohu určenú implementovanou knižnicou a zároveň polohu určenú GPS modulom ak je k dispozícii. Hlavná aktivita implementuje rozhranie `LocationSource.OnLocationChangedListener` a pomocou jeho callback metódy `void onLocationChanged(Location location)` zachytáva informácie o zmene polohy. Pri spustení aplikácie je vytvorený a spustený `OfflineLocationSource` z impementovanej knižnice ktorý zbiera informácie a určuje polohu pomocou mobilnej rádiovej siete a WIFI sietí, ktorú zobrazuje na mapu pomocou objektu `Marker`, ktorý má červenú farbu kvôli odlíšeniu. Zároveň je vytvorený `GoogleApiClient` a `LocationRequest` ktorí získavajú polohu z GPS modulu ak je k dispozícii a zobrazujú ju na mape rovnako pomocou ďalšieho objektu `Marker`, ktorý má však oranžovú farbu.

6.4 Zdieľanie nameraných dát

Aplikácia umožňuje v prípade dostupnosti informácii o GPS polohe zber dát pre ich možnosť zdieľania. Tieto dáta sú ukladané v `CSV` súbore a je možné ich pomocou aplikácie uploadovať určený na server, prípadne spracovať `CSV` súbor podľa vlastných potrieb. Forma dát v súbore je definovaná pomocou triedy `LogUploadCell` pre zájladňové stanice a `LogUploadWifi` pre prístupové body WIFI. Obe triedy implementujú rozhranie `Loggable`. Triedy nie je súčasťou implementovanej knižnice, ale aplikácie ku ktorej prislúchajú. Knižnica poskytuje iba nástroje na čítanie a zápis dát, no tvorba ich formy, ako v tomto je v réžii aplikácie podľa jej požiadaviek.

6.4.1 Upload dát základňových staníc

Namerané dáta o základňových staniciach sú zdieľané prostredníctvom servra www.opencellid.org. Jeden záznam v súbore obsahuje 7 informácií v nasledujúcom tvare:

```
latitude ; longitude ; mcc ; mnc ; lac ; cellID ; signalstrength
```

Súbor `CSV` obsahujúci dáta v takomto zložení je odoslaný na server pomocou verejnej metódy `static void uploadCellsMeasuredData(Context context,String filePath)`. Túto metódu je potrebné spustiť v samostátnom vlákne na pozadí, aby neblokovala hlavné vlákno aplikácie. Parameter `context` slúži na záverečné vytvorenie `Toast` správy ktorá informuje používateľa o úspechu, prípadne neúspechu uploadu dát a parameter `filepath` je cesta k požadovanému súboru. Súbor je odoslaný ako súčasť multipart HTTP POST requestu, ktorý obsahuje dva parametre:

- `key` - vygenerovaný API kľúč potrebný pre úspešnú komunikáciu so servrom `opencellid.org`, ktorý musí byť súčasťou každého requestu.
- `datafile` - dáta odosielaného súboru

Súbor je možné zdieľať alebo aj upravovať manuálne pomocou externých aplikácii na editáciu alebo zdieľanie súborov priamo v zariadení. Všetky takto vytvorené súbory sa nachádzajú v spoločnom adresári `OfflineLocation/upload_logs` ktorý je verejne dostupný v úložisku zariadenia.

6.4.2 Upload dát prístupových bodov WIFI

Namerané dáta o prístupových bodov WIFI sú uploadované pomocou služby Mozilla Location Service a jej API - `Geosubmit Version 2`. Odosielanie týchto dát má na starosti statická metóda `void uploadWifiMeasuredData(Context context, HashMap<Double,List<WifiInfoItem» wifiSets )`.

V prípade iba uploadu dát nie je potrebné generovať API kľúč[16]. Dáta sa odosielajú pomocou HTTP POST metódy v tele objektu JSON, ktorý obsahuje pole uložené pod kľúčom `items`. Súčasťou tohto poľa sú v tomto prípade tri položky pod nasledujúcimi kľúčmi [17]:

- `timestamp` - pod týmto kľúčom je uložená časová známka pozorovaných dát
- `position` - pod týmto blokom dát sa nachádzajú informácie o polohe, kde boli dáta namerané - `latitude` a `longitude`
- `wifiAccessPoints` - tento blok obsahuje ďalšie pole, ktorého jednotlivé prvky predstavujú konkrétne prístupové body WIFI. Každý prvok obsahuje dve hodnoty uložené pod kľúčami `macAddress` a `signalStrength`

7 Testovanie

Testovanie aplikácie prebiehalo v rôznych prostrediach pre porovnanie výsledkov, čiže v mestskej zástavbe, menej osídlených oblastiach a rovnako aj v skombinovaných úsekoch. Z každého merania je vypočítaný priemer a 95% inerval spoľahlivosti rozdielu vzdialenosti určenej polohy a skutočnej GPS polohy. Na každý súbor dát bola použitá Box-Coxová metóda transformácie dát, pretože pôvodné hodnoty nespĺňali normalitu. Po tejto transformácii mohli byť vypočítané korektné hodnoty priemeru a intervalov spoľahlivosti.

7.1 Presnosť lokalizácie pomocou rádiových sietí

Presnosť určenej polohy pomocou mobilnej rádiovej siete je podľa očakávania nie tak spoľahlivá ako poloha určená pomocou WIFI siete, kvôli veľkosti oblasti ktorú základnové stanice pokrývajú a kvôli nedostatku informácií o vyžarovacích charakteristikách týchto staníc.

Presnosť v mestskej zástavbe

- Meranie prebehlo na území Slovenskej Republiky v centre mesta Žilina. Trasa bola vykonaná chôdzou a viedla cez centrum mesta, mala približne 2.3 kilometra a bolo vytvorených 912 záznamov polohy z ktorých je určený nasledujúci výstup:

  - Priemer presnosti hrubo určenej polohy: 287,5 m
  - Určenie 95% intervalu spoľahlivosti: <272,5 m; 303,2 m>

  Toto meranie sa dá považovať za veľmi presné, nakoľko prebiehalo v centre mesta kde bol dostatok základňových staníc v okolí pomocou ktorých mohla byť presnejšie určená poloha.

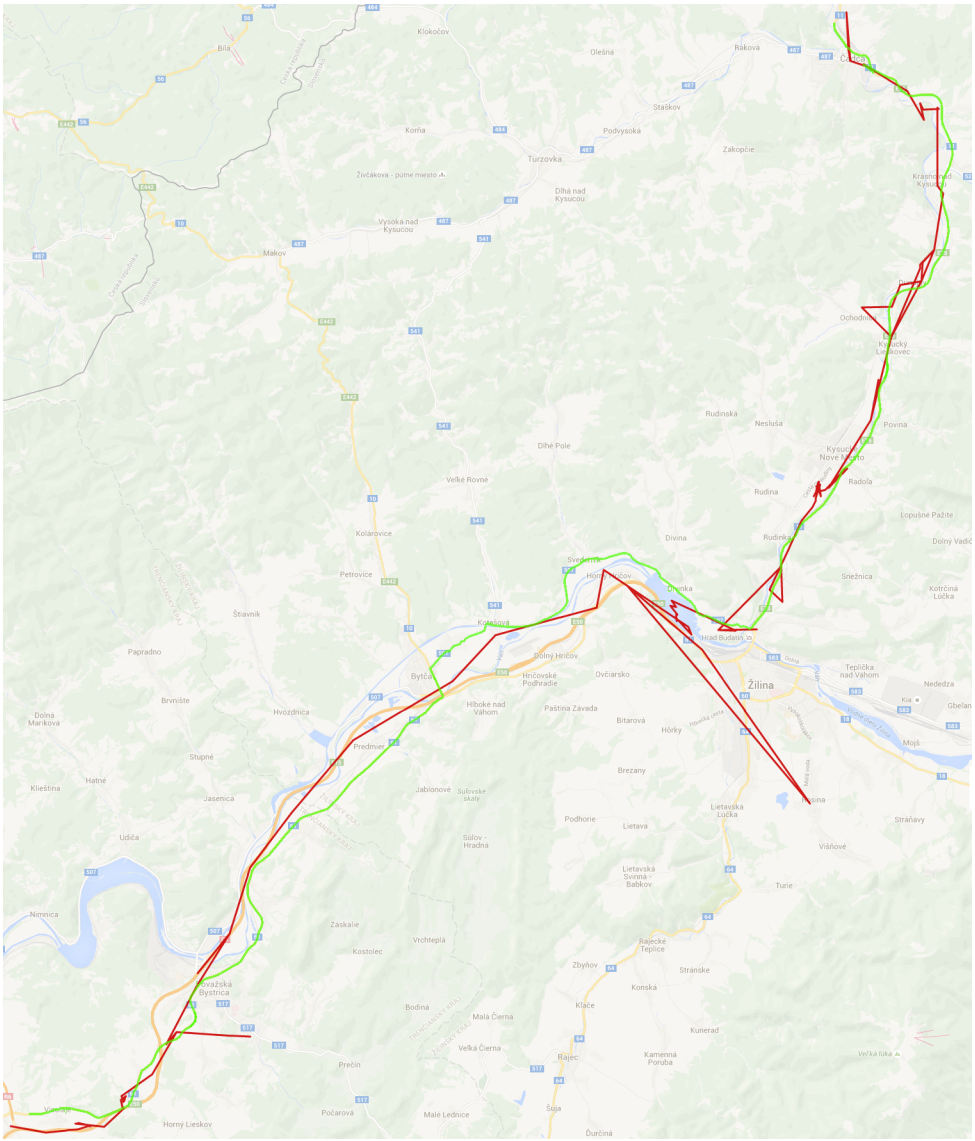
Presnosť v menej osídlenej oblastii

- Meranie prebehlo na území Slovenskej Republiky trase medzi mestom Čadca a Považská Bystrica. Trasa bola vykonaná osobným automobilom a viedla cez malé dediny za účelom merania presnosti hrubo určenej polohy v menej alebo vôbec neosídlenej oblasti, mala približne 70 kilometrov a bolo vytvorených 910 záznamov polohy z ktorých je určený nasledujúci výstup:

  - Priemer presnosti hrubo určenej polohy: 903,5 m
  - Určenie 95% intervalu spoľahlivosti: <858,9 m; 949,8 m>

  Počas tejto trasy bol zaznamenaný najvačší extrém určenej polohy od skutočnej zo všetkých testovacích meraní. V danom okamihu malo zariadenie informácie iba o aktuálne slúžiacej základňovej stanici, ktorá sa nachádzala vo väčšej vzdialenost a jeho poloha je

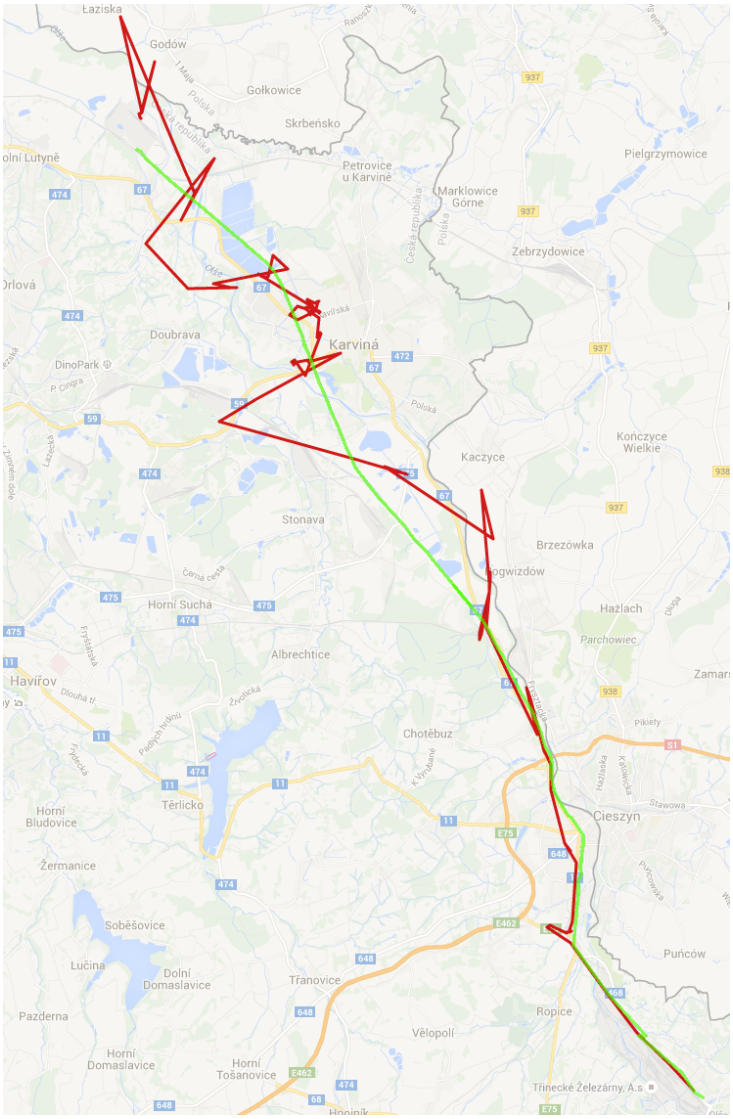
teda určená polohou tejto stanice, ako je vidieť na obrázku 15. Vzhľadom na možnosti pokrytia plochy stanicou je takáto situácia očakávaná, no spomedzi všetkých vykonaných meraní bol tento extrém jediný prípad.



Obr. 15: Ukážka trasy medzi mestami Čadca a Považská Bystrica.

- Meranie prebehlo na území Českej Republiky trase medzi mestom Třinec a obcou Dětmarovice, s prechodom cez mesto Český Těšín. Trasa bola vykonaná vlakovou dopravou a viedla cez mestskú zástavbu a zároveň aj cez menej osídlené oblasti a bolo vytvorených 400 záznamov polohy z ktorých je určený nasledujúci výstup:
  - Priemer presnosti hrubo určenej polohy: 667,2 m
  - Určenie 95% intervalu spoľahlivosti: <602,6 m; 735,7 m>

Ukážka tejto trasy vyznačenej na mape sa nachádza prílohách na obrázku 16. Z obrázku je vidieť rovnako isté výkyvy, no nie sú tak extrémne ako v predchádzajúcom prípade.



Obr. 16: Ukážka trasy medzi mestom Třínec a obcou Dětmarovice.

7.2 Presnosť lokalizácie pomocou WIFI sietí

Presnosť určenej polohy pomocou WIFI siete je podľa očakávania veľmi spoľahlivá a presná, oproti mobilnej rádiovej siete. Merania prebiehali v oblastiach s mestskou zástavbou.

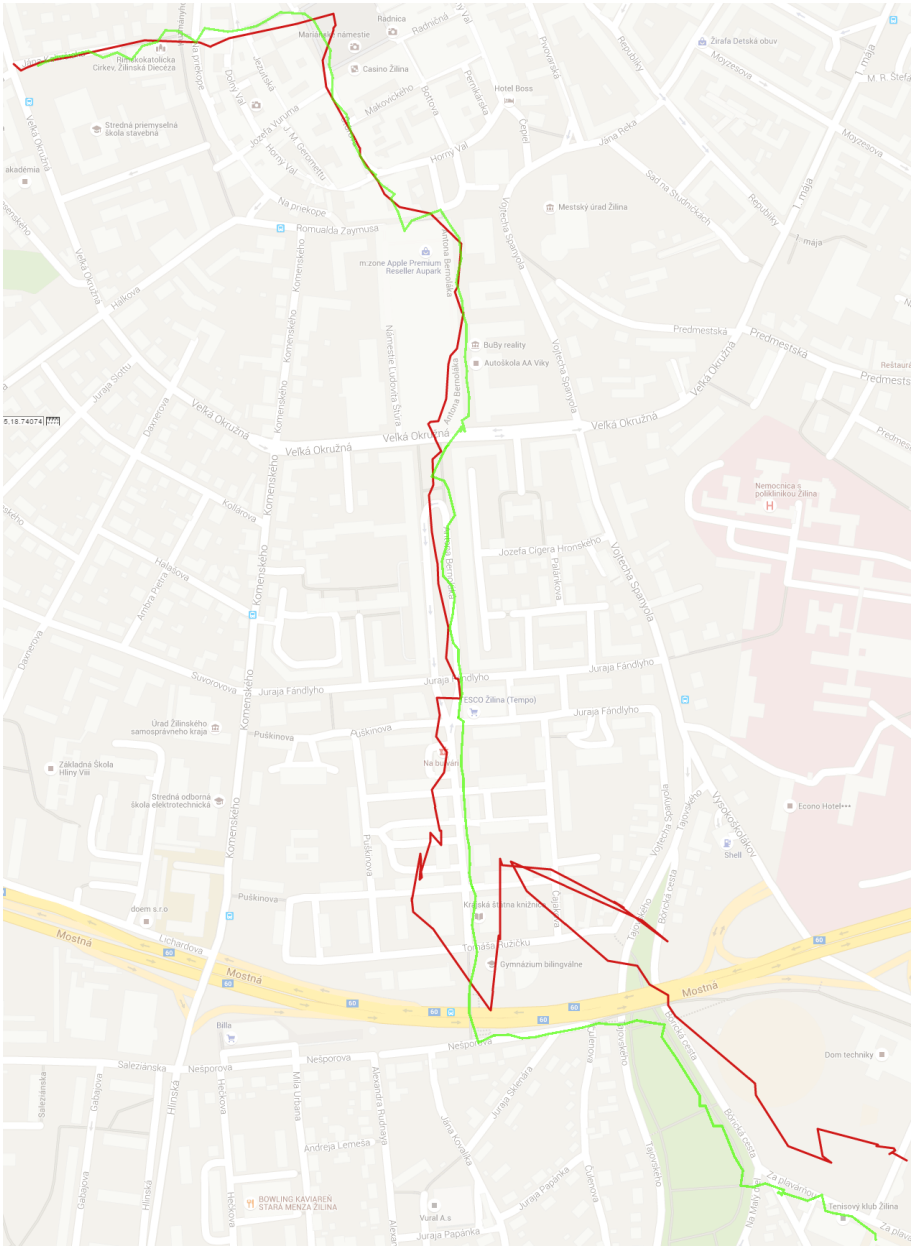
Presnosť v mestskej zástavbe

- Meranie prebehlo na území Slovenskej Republiky v centre mesta Žilina. Trasa bola vykonaná chôdzou a viedla cez centrum mesta, mala približne 2.3 kilometra a bolo vytvorených

595 záznamov polohy z ktorých je určený nasledujúci výstup:

- Priemer presnosti hrubo určenej polohy: 48,7 m
- Určenie 95% intervalu spoľahlivosti: <45,6 m; 52,0 m>

Ukážka tejto trasy vyznačenej na mape sa nachádza v prílohách na obrázku 17.



Obr. 17: Ukážka trasy v centre mesta Žilina.

Už z prvého pohľadu je jasné, že určená poloha pomocou WIFI siete je niekoľko násobne presnejšia oproti polohe určenej pomocou moblnej rádiovej siete. V tomto prípade tiež

vidieť mierny extrém, ktorý bol spôsobený prechodom oblasti, kde sa nachádzala rýchlostná cesta a zalesnený menší park. V tejto oblasti bol ešte zaznamenávaný slabý signál WIFI siete z ďalej stojacích bytových domov a budovy gymnázia, ktorý spôsoboval nepresnosť v určenej polohe.

Meranie v oblasti mimo mesta alebo mestskú zástavbu pre WIFI lokalizáciu počas testovania nebolo použiteľné. V drtivej väčšine prípadov poloha nebola vôbec určená kvôli úplnej absencii signálu z prístupových bodov WIFI, keďže nemajú dostatočný dosah. V občasných prípadoch sa polohu podarilo určiť, kedy som sa počas merania priblížil dostatočne blízko k oblasti so zástavbou kde bol patrný WIFI signál, no získané dáta ako celok neboli použiteľné, pretože vyzerali ako náhodne rozmiestnené body na mape, nebolo možné z nich určiť trasu. To je dôvod prečo v práci neuvádzam určenie polohy pomocou WIFI v oblasti mimo mestskej zástavby.

7.3 Zhrnutie výsledkov

Výsledky všetkých meraní sú rozdelené podľa oblastí - v mestskej zástavbe (intravillan) a mimo mestskej zástavby (extravillan). Ide o súhrnné výsledky všetkých meraní, kde je rovnako ako v ukážkach jednotlivých trás určený priemer, jeho 95% horný a dolný interval spoľahlivosti a namerané minimum a maximum vzdialeností určenej polohy od skutočnej pomocou GPS. Hodnoty sú zaokrúhlené na desatiny.

Mobilná rádiová sieť

Intravillan - zhrnutie výsledkov lokalizácie pomocou mobilnej rádiovej siete v zastavanej mestskej oblasti sa nachádza v tabuľke 4:

	Priemer	**Horný IS (95%)**	**Dolný IS (95%)**	**Max**	**Min**
Vzdialenosť (m)	252,6	262,5	243,0	1898,0	8,4

Tabuľka 4: Výsledky určovana polohy pre oblasti intravillan

Extravillan - zhrnutie výsledkov lokalizácie pomocou mobilnej rádiovej siete mimo zastavanej oblasti sa nachádza v tabuľke 5:

	Priemer	**Horný IS (95%)**	**Dolný IS (95%)**	**Max**	**Min**
Vzdialenosť (m)	816,9	847,2	787,3	11390,7	6,4

Tabuľka 5: Výsledky určovana polohy pre oblasti extravillan

Z tabuliek je jasne viditeľný rozdiel medzi týmito oblasťami, kde je určená poloha pomocou mobilnej rádiovej siete v mestskej zástavbe - intravillan, výrazne presnejšia ako v oblasti mimo mesta - extravillan.

WIFI sieť

Intravillan - Zhrnutie výsledkov lokalizácie pomocou WIFI siete v zastavanej mestskej oblasti - intravillan, sa nachádza v tabuľke 6:

	Priemer	**Horný IS (95%)**	**Dolný IS (95%)**	**Max**	**Min**
Vzdialenosť (m)	44,0	46,4	41,8	258,3	2,5

Tabuľka 6: Výsledky určovana polohy pre oblasti intravillan

Z tabuľky je zrejmé, že výsledky určenia polohy je možno považovať za veľmi presné, prihliadnúc na fakt, že samotný priemer je vo svojej podstate citlivý na extrémne hodnoty ktoré sa vyskytli aj v tomto prípade. Ako už bolo spomenuté, oblasti extravillan nie sú prezentované pre lokalizáciu pomocou WIFI siete, pretože vo väčšine prípadov v nich nie je dostupný signál, na základe ktorého by sa poloha mohla určiť.

7.4 Testovacie zariadenia

Výsledná aplikácia bola otestovaná na niekoľkých zariadeniach, aby bola jej funkčnosť potvrdená a prípadne odhalené chyby. Testovanie prebiehalo na nasledujúcich zariadeniach:

- `Sony Xperia Z3 Compact` - verzia operačného systému android 5.1.1
- `Asus Zenfone2` - verzia operačného systému android 5.0.1
- `Sony Xperia S` - verzia operačného systému android 4.1.2. Testovanie aplikácie na tomto zariadení demonštruje jej použitie na starších verziách API. V prípade tohto zariadenia nenastal problém pri získavaní informácií o susedných základňových staniciach.

7.5 Aplikácie s podobným zameraním

V obchode Play je k dispozícii mnoho aplikácii ktoré zisťujú polohu zariadenia, prípadne monitorujú polohu alebo prítomnosť základňových staníc v okolí mobilného zariadenia. Mnohé z nich však neurčujú polohu zariadenia ale slúžia skôr na monitorovanie. Medzi najznámejšie a najpoužívanejšie patria:

Google Maps Aplikácia od samotnej spoločnosti Google. Poskytuje množstvo kvalitných lokalizačných služieb a môže súžiť aj ako navigácia. Aplikácia ťaží maximum z možností Androidu a Google Services. Dokáže fungovať aj v offline režime s dátami uloženými v cache, určovať polohu pomocou GPS, ale aj mobilnej rádiovej siete alebo WIFI siete. Aplikácia poskytuje manuálne ukladať konkrétne oblasti, kedy sú stiahnuté potrebné dáta a daná oblasť je dostupná v offline režime. Dokáže zobrazovať detailné informácie o premávke alebo oblasti. V prípade určenia polohy pomocou mobilnej rádiovej siete alebo WIFI siete je určená poloha vo väčšine

porovnávaných prípadov veľmi podobná polohe určenej pomocou implementovanej knižnice popisovanej v tejto práci, čo sa dá považovať za veľmi dobrý výsledok.

G-MON Veľmi známa aplikácia ktorá slúži na monitorovanie mobilnej rádiovej siete alebo WIFI siete. Aplikácia obsahuje množstvo pokročilých možností na monitorovanie siete, vďaka ktorým umožňuje zobrazovať veľmi detailné informácie o sieti. Ponúka služby ako mapovanie prístupových bodov, takzvaný WarDriving, známy FieldTest alebo monitorovanie oblasti - Site Survey. Rovnako umožňuje import/export dát vo formátich CLF, CSV alebo KML. Dáta o základňových staniciach vo formáte CLF sú kompatibilné aj s implementovanou knižnicou a je teda možné ich medzi týmito aplikáciami zdieľať bez ďalších úprav. Aplikácia umožňuje zobrazovať na mape polohu jednotlivých základňových staníc aj v off-line režime, pomocou ktorých dokáže určiť polohu. Určená poloha pomocou mobilnej rádiovej siete alebo WIFI siete rovnako podobná a polohou určenou pomocou implementovanej knižnice. Všeobecne ide o veľmi kvalitnú aplikáciu.

OpenSignal Aplikácia slúži rovnako na monitorovanie signálu mobilnej rádiovej alebo WIFI siete. Aplikácia umožňuje vyhľadávať základňové stanice v okolí, zobrazovať pokrytie signálom na mape alebo určiť polohu zariadenia. Zaujímavou funkčnosťou aplikácie je ukážka smeru aktuálne slúžiacej bunky bez nutnosti otvárania mapy. Oproti predchádzajúcim aplikáciam má najkvalitnejšie spracované užívateľské rozhranie v ktorom sa jednoducho orientuje. Dokáže merať kapacitu, kvalitu, ping a rýchlosť používanej siete. Zaujímavou funkcionalitou je vizualizácia pokrytia oblasti signálom. Nevýhodou aplikácie je jej nemožnosť poskytovať svoje služby a pracovať v offline režime.

WIGLE Wifi Wardriving Aplikácia slúži ako klient pre server www.wigle.net, ktorý zbiera dáta o WIFI sietiach a na ich základe určuje polohu zariadenia - vardriving. Aplikácia umožňuje namerané dáta odosielať na server, prípadne s nimi ďalej pracovať prostredníctvom exportovaných súborov. Aplikácia umožňuje aj zobrazovanie štatistických výsledkov zoradených podľa časového obdobia jej používania. Uložené dáta umožňuje exportovať do súborov v CSV alebo KML formáte, prípadne ich importovať. Určená poloha touto aplikáciou pomocou WIFI siete je ako v predchádzajúcich prípadoch veľmi podobná s polohou určenou pomocou implementovanej knižnice.

8 Záver

Cieľom tejto práce bolo vytvoriť rozšíriteľnú knižnicu ako offline lokalizáciu pre platformu Android, vytvoriť ukážkovú aplikáiu ktorá by túto knižnicu využívala a otestovať ju na reálnych zariadeniach. Zadanie práce som splnil a aplikáciu otestoval v rôznych prostrediach. Výsledky určenia polohy pomocou mobilnej rádiovej siete sú podľa očakávania menej presné ako určenie polohy pomoco WIFI siete. Dôvodom je nedostatok informácií o základňových staniciach a ich oblasti pokrytia, ktoré operátori nezverejňujú. Viditeľný rozdiel v presnosti je tiež ovplyvnený prostredím v ktorom sa mobilné zariadenie nachádza. V mestskej zástavbe s vyšším počtom základňových staníc je určená poloha presnejšia ako v oblasti s menšou hustotou zástavby, prípadne žiadnou, kde je počet a rozmiestnenie základňových staníc menšie. Otestované boli aj možnosti zdieľania monitorovaných dát o rádiových základňových staniciach alebo prístupových bodoch WIFI, vďaka ktorým je možné prispieť k zlepšeniu kvality verejných zdrojov o polohe týchto prvkov.

Knižnica je použiteľná na zariadeniach s verziou systému Android 4.0 Ice Cream Sandwitch a podporuje teda aj staršiu verziu Telephony API, v ktorej sa vyskytuje problém s nemožnosťou získať informácie o susedných základňových staniciach. Tento problém sa týka samotného operačného systému a v prípade jeho výskytu nie je teda možné ho obísť. Pri implementácii knižnice som sa snažil zachovať jednoduchosť použitia knižnice ako výsledného produktu, tak aj jej možnú úpravu alebo rozšírenie pomocou dodržiavania princípov objektovo orientovaného programovania. Knižnicu je teda možné doplniť o ďalšiu funkcionalitu, prípadne upraviť jej jednotlivé komponenty jednoduchým spôsobom. Počas implementácie som sa nestretol so žiadnym významným problémom okrem spomínanej chyby nemožnosti zistiť informácie o susedných základňových staniciach.

Rozšírenie knižnice do budúcna je možné napríklad v doplnení o spracovávanie bluetooth dát zo stále sa rozširujúcich zariadení beacon, ktoré by prispeli k univerzálnosti jej použitia. Priestor na rozšírenie funkcionality je tiež v obšírnejšom využití získaných dát pri rôznych lokalizačných službách alebo monitorovaní siete.

Peter Gorčák

Literatúra

[1] AHONEN, Suvi, Sofoklis KYRIAZAKOS, Jaakko LÄHTEENMÄKI, Raffaele MENOLASCINO a Seppo PARKKILA, LAITINEN, Heikki (ed.). *Cellular network optimisation based on mobile location: Cellular Location Technology* [online]. 2001 [cit. 2016-02-21]. Dostupné z: https://lyle.smu.edu/~rajand/courses/8377/papers/e911_recent.pdf

[2] SINGH, Balaram, Soumya PALLAI a Susil Kumar RATH. *A Survey of Cellular Positioning Techniques in GSM Networks* [online]. , 6 [cit. 2016-02-21]. Dostupné z: https://www.researchgate.net/publication/256736858_A_Survey_of_Cellular_Positioning_Techniques_in_GSM_Networks

[3] CHEN, Chi-Chang, Chi-Yu CHANG a Yan-Nong LI. Range-Free Localization Scheme in Wireless Sensor Networks Based on Bilateration. In: *Hindawi Publishing Corporation* [online]. 2012 [cit. 2016-02-21]. http://www.hindawi.com/journals/ijdsn/2013/620248/

[4] *Mobile Positioning Techniques in GSM Cellular Networks: A Comparative Performance Analysis* [online]. 2012, 2(6) [cit. 2016-02-21]. Dostupné z: http://www.ijctee.org/files/VOLUME2ISSUE6/IJCTEE_1212_05.pdf

[5] Timing Advance In LTE. TELETOPIX.ORG: *Telecom Techniques Guide* [online]. [cit. 2016-02-21]. Dostupné z: http://www.teletopix.org/4g-lte/timing-advance-in-lte/

[6] Základní lokalizační metody v GSM.3 *Access Server:* Telecom Techniques Guide [online]. [cit. 2016-02-21]. Dostupné z: http://access.feld.cvut.cz/view.php?cisloclanku=2006022801

[7] TelephonyManager. *Android Developers* [online]. [cit. 2016-02-07]. Dostupné z: http://developer.android.com/reference/android/telephony/TelephonyManager.html

[8] android.location. *Android Developers* [online]. 2015 [cit. 2016-02-07]. Dostupné z: http://developer.android.com/reference/android/location/package-summary.html

[9] Google Maps Geolocation API: Get a Key/Authentication. *Google Developers* [online]. [cit. 2016-02-29]. Dostupné z: https://developers.google.com/maps/documentation/geolocation/get-api-key

[10] The Google Maps Geolocation API: Google Maps Geolocation API. *Google Developers* [online]. [cit. 2016-02-29]. Dostupné z: https://developers.google.com/maps/documentation/geolocation/intro

[11] NeighboringCellInfo. *Android Developers* [online]. [cit. 2016-02-07]. Dostupné z: http://developer.android.com/reference/android/telephony/NeighboringCellInfo.html

[12] New clf format in G-MoN 3.3.0. *Gmonclf4* [online]. [cit. 2016-02-07]. Dostupné z: `https://sites.google.com/site/clfgmon/`

[13] LocationManager. *Android Developers* [online]. [cit. 2016-02-07]. Dostupné z: `http://developer.android.com/reference/android/location/LocationManager.html`

[14] AsyncTask. *Android Developers* [online]. [cit. 2016-03-20]. Dostupné z: `http://developer.android.com/reference/android/os/AsyncTask.html`

[15] Transmitting Network Data Using Volley. *Android Developers* [online]. [cit. 2016-02-07]. Dostupné z: `http://developer.android.com/training/volley/index.html`

[16] Services API: API Access Keys. *Inchaea 1.5 documentation* [online]. [cit. 2016-04-06]. Dostupné z: `https://mozilla.github.io/ichnaea/api/index.html#api-access-keys`

[17] Geosubmit Version 2. *Inchaea 1.5 documentation* [online]. [cit. 2016-04-06]. Dostupné z: `https://mozilla.github.io/ichnaea/api/geosubmit2.html`

A Obsah priloženého CD

- PDF súbor s textom práce: `diplomova_praca.pdf`
- Projekt v programe Android Studio vo rormáte .zip: `projekt.zip`
- Vytvorený APK súbor s aplikáciou vo formáte .zip: `aplikacia.zip`